Object Learning and Recognition using Human-Guided Object Segmentation and Active Vision

Matthijs Zwinderman
June 22, 2009

Master Thesis
Artificial Intelligence
Dept. of Artificial Intelligence,
University of Groningen, The Netherlands

Internal Supervisors:
Gert Kootstra, Ph.D Student, Artificial Intelligence, University of Groningen
Prof. Dr. Lambert Schomaker, Artificial Intelligence, University of Groningen
External Supervisor:
Paul E. Rybski, Ph.D., Systems Scientist, The Robotics Institute, Carnegie Mellon University

university of groningen

Abstract

Humans are remarkably apt at segmenting objects in complex natural scenes and they are able to learn new objects quickly (Feldman, 2003). As robots will become more integrated in the workflow of humans, they should have such skills as well (Breazeal et al., 2004, Brooks and Breazeal, 2006). The concept of objects (e.g. what is an object, what are the boundaries of an object) is useful for, but not limited to, human-robot collaboration and task learning (Breazeal et al., 2004, Gopalakrishnan et al., 2005, Nicolescu and Mataric, 2001). However, it is hard to define what exactly constitutes an object as the meaning of the word "object" is very broad and dependent on semantics and context (Feldman, 2003). We propose the use of human knowledge of objects to segment objects in space as a first step to object learning. We also propose a method for a mobile robot to use this segmentation in combination with stereo vision to create a segmentation in three-dimensional space. This segmentation is then used as a basis for a learning method using active vision. The system was tested on a mobile robot with an extensive dataset and it is shown that the use of human guided object segmentation results in a better recognition rate.

A natural way of referring to an object is by using a deictic gesture like pointing (Brooks and Breazeal, 2006). A deictic gesture is a gesture dependent on context and environment. A simple gesturing device, the laser pointer, is proposed to aid in using the natural deictic gesture of pointing to guide the robot's attention a new object. After having achieved joint visual attention in this manner, the use of the laser pointer is extended from referring to the object, to creating an initial object segmentation. The laser detection technique uses a novel approach to discard false detections and human error, by assigning each detection a position in three-dimensional space using stereo vision and performing clustering. This initial segmentation is optionally enhanced with the use of a computer vision technique called GrabCut (Rother et al., 2004). Stereo vision is used to calculate the object's dimensions in three-dimensional space.

The method is implemented on a mobile robot, equipped with a stereo camera. Object referral and segmentation is tested on a dataset of 7 real-world objects, in various positions from the robot and with two backgrounds. It is shown that the segmentation techniques perform very well.

The object representations are created by combining SURF (Speeded Up Robust Features) feature extraction (Bay et al., 2006) with stereo vision. All features that are not within the object's dimensions as calculated in the segmentation step, are discarded. The remaining features are filtered for robustness using active vision, by letting the robot drive around the object. By using the knowledge of the object's dimensions and position, the system is capable of automatically tracking the object while driving around it, making it possible to follow any path around the object.

The learning and recognition methods were tested on a database of 21 objects, recorded in a real-world office environment. The objects vary greatly in size, shape, color and texture. This dataset is considerably larger than used in similar research. It is shown that by filtering the keypoints using the human segmentation and active vision, the number of keypoints can be greatly reduced while not decreasing recognition accuracy. It is further shown that object recognition works reasonable in highly complex real-world environments, with lighting changes and object occlusions, even when using only one view of the scene.

Acknowledgement

This thesis is the finishing touch of many months of research done to fulfill the requirements for the degree Master of Science in Artificial Intelligence at the University of Groningen. Many of those days went by programming, processing data, and writing my thesis. But luckily some time remained for meetings with my advisors Gert Kootstra and Paul Rybski, to whom I am grateful for their professional guidance and near unlimited enthusiasm.

I have had the great pleasure of studying abroad because of this research and I have found my stay in the United States of America very pleasant. The experience has greatly widened my horizon. The Carnegie-Mellon University (CMU) proved to be a great environment and I enjoyed visiting many interesting talks and meeting with some great minds within robotics.

During my stay at CMU there have been several other people that were of tremendous help in supplying me with ideas and helping me in any other way possible. I would especially like to thank the two other Dutch graduation students who's help and friendship was not limited to just research: thank you, Jaldert Rombouts and Tijs Zwinkels!

Furthermore I would like to acknowledge the efforts of Brandi, my family, my parents and brother especially, and my friends in trying to keep me sane during the writing of this thesis.

Thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Objects are at the base of how humans perceive the world (Feldman, 2003) and as the intensity of human-robot interaction can be expected to increase in the future (Breazeal et al., 2004), it is important that robots have knowledge about objects as well. However, the object concept is not easy to grasp which makes it hard to create a solid definition of a generic object (Feldman, 2003). Without a good definition of an object it is hard to create a computer-vision or robot system that can automatically detect and learn generic objects. The amount of objects in the world is infinite, which means that it is impossible to create an *a priori* dataset, let alone maintaining it or having a robot use it in real time.

We now have an apparent paradox: humans perceive the world in objects, making it necessary to incorporate the concept of objects in robot systems as well, but this concept is so broad and undefined that it is not currently possible to create a useful description of it nor is it possible to create a finite database of all objects. It seems that what we have decided is necessary, is unattainable. But the paradox can be solved by using the human knowledge that is readily available to us in the case of human-robot collaboration. This makes it necessary to have a method for the user to teach a novel object which is appropriate for a task to the robot *on the fly*. This method should be unambiguous and natural for the human, but should also be robust when implemented on a mobile robot. The goal of this thesis is to create such a method and to implement this in such a way, that the resulting object representation is useful for the recognition of a large group of objects in a real-world office environment.

Several other systems have been proposed where objects can be taught to a robot with human assistance (Fitzpatrick (2003), Ghidary et al. (2002), Lomker and Sagerer (2002), Moller et al. (2005), Steil et al. (2007), Toptsis et al. (2004), and others, see also chapter 2). The first step of teaching an object, is making clear which object is going to be taught. The human needs to accomplish *joint visual attention* with the robot, meaning that the attention of the robot should be focussed on the object that the human has in mind (Brooks and Breazeal, 2006). This can be accomplished in several ways, by using speech or by using gestures, or a combination of these (see section 2.2). As understanding speech requires knowledge of the scene and the object beforehand, it is less suitable for object learning (Kemp et al., 2008). Gestures do not have this problem, but natural gestures are by nature inaccurate and recognition techniques add another layer of inaccuracy (Brooks and Breazeal (2006) and Ziegler et al. (2006), see section 2.2.1). This has led to the creation of several assisting devices, which make resolving the gesture easier. To increase the accuracy of object designation such an assisting device is used here as well. However, instead of using a complicated device which can only be used in special environments (as used by Wilson et al. (2003) and Patel and Abowd (2003), see also section 2.2.2), a simple and natural to use pointing device is used: a laser pointer. The approach used here is based on Kemp et al. (2008), but instead of merely using the laser pointer as object designation the method is extended so that objects can also be segmented by the system using the trail left by the laser. The laser detection method is tested on several types of objects, in different environments.

After the human teacher has designated the object with the laser pointer the robot begins to create an object representation, but this representation should not contain information from the background. In similar systems user information is used to distinguish the object from the background (for instance Ghidary et al. (2002), Moller et al. (2005), Steil et al. (2007), see also section 2.3.1) or the robot is

presented with a simplified world so it can segment the object on its own (Fitzpatrick (2003), Gopalakrishnan et al. (2005), Kootstra et al. (2007), see also section 2.3.2). In this work user information given with the laser pointer is combined with a segmentation technique from computer-vision called GrabCut (Rother et al., 2004).

Object representations come in many different variations, based on shape or appearance (Dutta Roy et al., 2004), several of these representations can be found in section 2.4. From these methods SURF is the most promising, it is robust against various transformations and distortions, while it is optimized for speed compared with similar methods (Bay et al., 2006). This is why SURF is used in the current research to store the object representations and match objects in the recognition phase.

Besides simply storing features from one viewpoint of the object, the robot platform offers the possibility of exploring an object by driving around it. Using an active vision approach based on Kootstra et al. (2007) a robust object representation is created by circling the object. Using active exploration has several advantages, one can filter features based on their stability (by checking their robustness against changes in viewpoints) making the object model smaller. Because features are recorded from multiple viewpoints, objects can be recognised in several poses resulting in a better overall recognition rate (as evidenced by results of research as found in 3.2.2). An adaptation is made to the method of Kootstra et al. (2007) so that less viewpoints are necessary to create a similar object model. Instead of relying on a circular path to filter background features from becoming part of the object model, the human segmentation is extended to a third dimension so that this filtering can be done using stereo vision and odometry information. This results in a method that does not rely on a circular path, making it possible to create an object model when it is impossible to perform a full circle around the object due to obstruction from other objects on the path. It is now also possible for other objects to be close to the target object without features from this object being learnt, it is even possible that such objects partially obscure the target object. Although currently the robot is controlled by the experimenter, the methods described in this work can be used for an autonomously driving robot as well.

To test the system an extensive dataset was created with 21 objects and both learning and recognition was done in real-world settings. In addition to this, the method was tested on a dataset containing scenes with multiple objects, some being partially obscured. The objects differ in size, color and structure. This makes the results of this research quantifiable to situations which the robot might encounter when actually performing tasks in collaboration with humans. The size of this dataset is much larger than that of similar research, where the amount of objects in a set differs from 2 (Zickler and Veloso, 2006), 4 (Ekvall et al., 2006), 7 (Kootstra et al., 2007) and 9 (Andreasson and Duckett, 2003).

This thesis is organized as follows: related work is described in chapter 2, the algorithms and general methodology are described in chapter 3, the experimental results are detailed in chapter 4, and finally the approach is discussed in chapter 6.

Chapter 2

Theoretical Background

The focus of this work is to create a system with which an object can easily be taught to a robot companion in a complex environment, with only limited user involvement. In this case the robot learns such an object representation by actively exploring the object.

It can be expected that the use of robots in human environments will increase and it is important that the complexity of these environments pose no problem to these robots, especially when robots are to function in together with humans (Breazeal et al., 2004). In the case of human-robot collaboration, where humans and robots are trying to accomplish spatial tasks involving objects together, it is important for robot and human to develop a spatial common ground (Brooks and Breazeal, 2006). This means that the robot needs to have a similar understanding of the environment as the human has, it has to be able to use the same concepts that humans use.

This work focusses on teaching specific objects to robots to develop such a spatial ground, as objects are a very important concept for humans to describe their world (Feldman, 2003). The use of an object concept is not limited to human-robot collaboration. It can also be used for task learning, where a human or robot teaches a task to another human or robot by making use of demonstration using physical objects or by using dialog to explain a task using object concepts (Breazeal et al., 2004, Nicolescu and Mataric, 2001). Another use for the object concept is autonomous semantic structure recognition (Ekvall et al., 2006), where a robot learns about a specific room by examining the objects in it, for instance, recognize a kitchen if there are several food-items and food-processing units in the room. And lastly localization, where the robot recognizes a specific location by locating objects in it (Gopalakrishnan et al., 2005).

The rest of this chapter is divided in a further introduction to the object concept (section 2.1), object referral (2.2), segmentation (section 2.3), learning object representations and using these for object recognition (section 2.4).

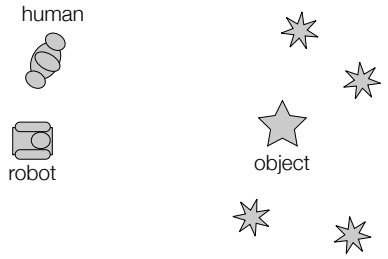


Figure 2.1: An example set-up where human-guided object learning can be used. The target object is surrounded by background objects. Using the method described in this research the human can easily refer to this object and let the robot create an object representation.

2.1 The object concept

"ob•ject: *A material thing that can be seen and touched*"

- New Oxford American Dictionary (*2nd Edition*)

As described by Feldman (2003), objects play a central role in understanding human cognition in cognitive science. For instance, objects are the building blocks to the concept of a physical world for children. Furthermore, researchers have shown that visual attention is delineated by object boundaries (Feldman, 2003, Olson, 2001). The object concept can be found as an influence in even the earliest stages of neural processing in the visual cortex; by using experiments with visual illusions it is clear that neurons with small receptive fields, in the early cortical visual areas are involved in considering which local features belong to which object (Olson, 2001).

Defining the object concept is notoriously difficult, besides being spatially coherent bundles in an image plane, one can think of objects as being units of ontology or as things with which we interact in a physical way. These different ways of looking at objects are connected, but logically independent of each other (Feldman, 2003). Even when limiting the concept to the first case, an object being a coherent bundle of visual information, it is hard to create an object definition as no single visual grouping cue (such as closure, connectiveness, regularity of shape and so forth) defines objects (Feldman, 2003). An approach to solve this is to use the fact that perceptual organization is hierarchical, a notion found in psychology (Feldman, 2003) and computer vision (Marr and Nishihara, 1978). Instead of using one single visual grouping cue, multiple cues can be combined in a tree, combining information. Such hierarchical organization can be found in many computer-vision segmentation techniques (see section 2.3). Another difficulty in defining the object concept is that it relies on much more then properties in the image plane (Feldman, 2003, Treisman, 1986). Such properties are both semantic and contextual, an example is grouping of visual features: whether a group of visual features is an object by itself or is merely part of an object relies on the task at hand and on the environment.

Conclusion

The object definition crosses many domains such as ontology, mental dynamics and human perception, and the scope of what is referred to as an object relies on context. Because of this, creating a global definition of an object with which a robot can autonomously segment an object reliably, lies outside of the scope of the current research. Fortunately, it is unnecessary to define such a broad object concept, as knowledge about objects and the context is already available in the form of a human collaborator in this research. By utilising the human-robot interaction the robot does not need to have a full understanding of a broad object concept. Instead the human can act as a teacher to aid with guiding the robot's attention to an object and create a preliminary segmentation it, before the robot creates a representation of it.

2.2 Referring to objects

In human-robot interaction it is often necessary to refer to a specific object. A distinction can be made here between concretely referring to an object ("screw no. 14") or by using space deixis ("that thing"). The last category (also called *place deixis*), stems from the more general deixis (also called *indexicality*), the use of such expressions as *you, here, now* (Horn and Ward, 2004). For deictic referring, an *origo* (Bühler, 1934), a *referent* and an *addressee* (Horn and Ward, 2004) are necessary, the origo being the starting point for the deictic reference, the referent being the object or person referred to and the addressee being the person(s) listening. In space deixis, the use of gestures is an obvious way to secure the addressee's attention on the referent and it has been thought crucial for acquisition, teaching new objects and concepts to children (Horn and Ward, 2004). Research shows that over 50% of spontaneous gestures are concrete deictics, gestures referring to a specific object (Gullberg, 1999). One of the problems with spatial deixis is that context is very important. Consider the example of Horn and Ward (2004) (based on Quine (1961)), where a referrer is pointing at a river while saying "This is the Thames": the referrer could mean the water, the left bank of the river or the sun sparkling on the water. The referrer has to

keep the addressee's frame of reference in mind: based on the current position and the task at hand, the referrer has to pick out a gesture based on what (s)he thinks the addressee will expect to be picked out (Horn and Ward, 2004). Resolving spatial deixis thus highly depends on an understanding of the environment and context.

Because deictic references come natural to humans, solving deictic references is also a research goal in robotics, especially in human-robot interaction. The importance of this is made more significant by Brooks and Breazeal (2006): solving deictic references are part of creating *joint visual attention*, which is necessary for humans and robots working together on real-world spatial tasks involving objects. Due to the complexity of resolving deictic references and because the term is very general, the meaning of deictic references is simplified in robotics research. One simplification is made by Brooks and Breazeal (2006), where deictic referral is limited to concrete deictics to specific objects in the visual field (versus referring to persons and concepts, and lexical deixis). The same limitation is made in most other robotics research described here as well, combined with other limitations which are in part due to the manner of implementation.

Several methods of resolving natural pointing have been suggested in the robot domain, which will be discussed in section 2.2.1. Due to the problems with resolving natural gestures several assisting devices have been researched, which are discussed in section 2.2.2.

Conclusion

Concrete deictic references to objects are natural for humans and are a fundamental aspect of human-robot interaction. The implementation of reference solving is a hard problem, as deictic references depend on an understanding of the environment and the current context. It is possible however, to simplify the definition of deictic references in this domain to make the problem easier.

2.2.1 Natural means of referring

The two natural means of referring to an object are concrete deictic gestures (Brooks and Breazeal, 2006) and speech (Wachsmuth et al., 2000), and combinations of these methods. Especially the latter category is interesting, as 90% of gestures occurs in conjunction with speech (McNeill and Arnheim, 1996). Both speech and gestures have a large error and neither offers a solution for object segmentation as the object concept needs to be known before hand, the advantage of is that these methods are natural to humans. Besides pointing, there are several other deictic gestures, such as nodding, pointing, a directed gaze or even in some cultures pursing ones lips (Horn and Ward, 2004). In the research discussed below a concrete deictic gesture means pointing.

Steil et al. (2007) use yet a different approach to refer to a specific object, by having the human present the object to the robot by holding it in front of the robot's camera. Using stereo vision and a color model to subtract the user's hand the object can successfully be segmented and designated. The approach is limited to small objects that can be hand carried, however, and both speech and pointing do not have this limiting problem.

When pointing at something, there is a difference between the *demonstratum*, the position that is pointed at, and the referent (Brooks and Breazeal, 2006). This could occur due to parallax, or the human desire not to obstruct the robot's view (Brooks and Breazeal, 2006). On top of this, recognition robustness is an issue, research suggests gesture estimation errors of 22.7° for the upper arm and 25.7° for the lower arm (Ziegler et al., 2006). As the distance increases between referrer and referent, the absolute error becomes larger. Consider the error of 25.7° for the lower arm: even if the upper arm position is completely known, the absolute error could be up to 96 centimeters at a distance of two meters. Because of this, much of the research on detecting natural gestures limits the proximity of the human to the referent (Ghidary et al., 2002, Lomker and Sagerer, 2002, Moller et al., 2005, Toptsis et al., 2004). This solves another problem inherent to pointing, the lack of a distance argument in pointing (Brooks and Breazeal, 2006). This limitation comes at a great cost, the human needs to walk up to each object before being able to refer to it. Other limitations can be made on types of gestures and on the environment, for instance the clothing of the human (Ghidary et al., 2002).

Using speech-only systems has the problem that the robot needs to have knowledge about the object and the scene before a human can refer to them (Kemp et al., 2008), making these unsuitable for teaching objects to a robot. Speech systems are often limited in the range of human input they can process to make recognition more computationally tractable. This means that the user would need to limit speech to a set of commands (Tomko and Rosenfeld, 2004, Yoon and Rybski, 2007). As such a set has to be learned, the approach becomes less natural to the human.

Systems combining speech and gestures are often used in human-robot interaction, because of the limitation of both individual methods and because gestures are accompanied by speech in most cases. The dialog management system of Bielefeld Robot Companion (BIRON) is an example of such a system (Moller et al., 2005, Toptsis et al., 2004). With this system the user can choose from a limited set of words and sentences and optionally combine them with gestures. The dialog system can also handle spatial relations between objects and colors, it can learn new color models on the fly.

Another such system was described in Ghidary et al. (2002), here an object is referred to by a human by speech and a deictic gesture. The human teacher has four different ways of teaching an object to the system, it can point to the centre of an object, to two corners of an object or describe the object's position relative to the robot or other learnt objects. For each of these methods, except for the second, the user needs to explicitly specify the size of the object by speech.

Lomker and Sagerer (2002) also propose a system combining speech and gestures. The human can give size and color constraints using simple verbal commands and can point at an object. The type of the objects that can be learnt is limited to small table top items and currently the system is not implemented on a robot. For all three of the previously discussed systems the human teacher needs to be in close proximity to the object.

Conclusion

Natural pointing gestures are inaccurate, there is an error between what is pointed to and what is meant with the gesture and an understanding of the environment and the context is necessary to resolve these gestures. Current implementations of gesture recognition on robot systems only work on a limited set of gestures and impose a severe restriction on the distance over which an object can be pointed at. In addition, these natural methods do not offer a way of segmenting objects from the background. However, in combination with speech understanding these systems do offer the possibility of a natural means of accomplishing joint visual attention between human and robot. Due to the many practical issues that still remain with natural means of object referral, researchers have developed systems that are more tractable which will be discussed in the next section.

2.2.2 Assisted means of referring

Some systems try to replicate the natural pointing behavior or extend this behavior, others replace the natural gesture all together. One type of replication is to use an intermediary representation on a screen like a PDA (Lundberg et al., 2003) or a computer screen (Kemp et al., 2008, Leroux et al., 2007). The two main disadvantages of this method, are that the creation of the intermediate representation is problematic and that the interaction between the user and the robot becomes considerably less natural.

There are also devices that extend natural pointing while making the recognition of the pointing easier. Some of these require an *intelligent environment*, while others can work in more general environments. One assisted pointing device that can work in general environments is the laser pointer, it will be discussed in the next section (2.2.2), some other extensions on natural pointing will be discussed first.

Two advanced examples of extensions of natural pointing are the XWand (Wilson et al., 2003), see figure 2.2, and its derivative the World Cursor (Wilson and Pham, 2003). With the XWand the user can select objects in a room and interact with these objects. To accurately detect where a user is pointing, the system uses a combination of the built-in sensors (a gyroscope, a magnetometer and an accelerometer) and two infra-red cameras which have to be present in the room to detect the built-in infra-red LED light. There are several limitations to this system, it only works in a specially equipped room and in some indoor environments the magnetometer gives incorrect measurements meaning that orientation calculation fails. Interaction with objects is only possible if these objects are connected to

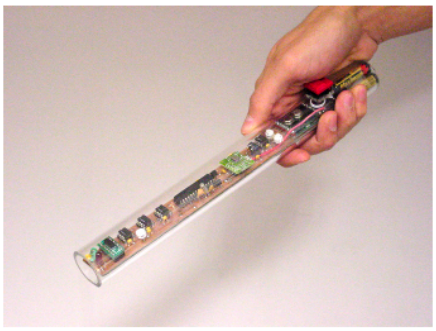


Figure 2.2: The XWand, a complex device which can be used as an extension of natural pointing. Image from (Wilson et al., 2003)

the system. The World Cursor aims to solve some of these limitations (Wilson and Pham, 2003), it is an extension on the XWand. Instead of two cameras the system uses a ceiling mounted laser pointer. When the user changes the orientation of the XWand the system moves the laser pointer and the user can see to which object the system is currently pointing. If the geometry of the room is known, a 3D position of the laser dot can be extracted by combining the yaw and pitch of the laser pointer with structural information of the room and the object can be selected and interacted with. This system removes the need for an exact position estimation of the XWand and adds extra visual feedback of selection to the user in the sense of a laser dot. However, the system still relies on a complicated pointing device and only works with objects that are compatible with it. The system requires knowledge of the world it is operating in and the room needs a laser pointing system attached to the ceiling.

Another pointing device that requires an intelligent environment is that of Patel and Abowd (2003). The pointing device itself is much simpler than the XWand or World Cursor and the system is not limited to one specific environment and unlike the previous systems, the researchers expect it to work in both indoor and outdoor environments. Patel and Abowd (2003) use a normal cellphone to which a laser pointer is attached as pointing device. With the device the user can activate *active tags* on objects, which are small devices with phototransistor, a micro controller and wireless connectivity. The active tags send a signal back to the cellphone which processes the signal and takes the appropriate action. The system is less complicated as the XWand based system and the restrictions on the environment are less severe. But the limitation of the system is that only objects that are specifically tagged can be used and that the pointing device, although less complicated, remains complicated.

Laser Pointer

Instead of the complicated pointing devices discussed in the previous section, a simpler extension of natural pointing is possible. Laser pointers have been and are being used during presentations and meetings to direct human attention as an extension of normal pointing (Kirstein and Mueller, 1998, Olsen Jr and Nielsen, 2001). The use of the laser pointer is not limited to this domain, it has also proven to be useful for helper animals for the disabled, like monkeys and dogs (Helping Hands association, 2009). In the last case, a human uses the laser pointer to direct the attention of the animal to a specific object and can issue a command so that the helper animal interacts with the object.

This has inspired Kemp et al. (2008) and Kazi et al. (1995) to use a laser pointer for human robot interaction. The advantage of a laser pointer in combination with a robot with camera, is that it does not require models of the environment or a specialized pointing device and there is no need to label objects beforehand. All that is needed is a simple, off the shelf laser pointing device. The added benefit of this method is that it provides a clear visual feedback to the human.

The first system combining the laser pointer with robotics research, the “Multimodal User-Supervised Interface and Intelligent Control (MUSIIC)”-system of Kazi et al. (1995), works only for a small table top objects, as the robotic system was limited in its movement. In contrast, the system described by

Kemp et al. (2008) uses a mobile robot, showing that such an approach works in larger environments and more objects as well. They report detection-rates of 99.4%, even for small objects such as bottles and phones located up to three meters from the robot. After detecting the laser pointer, this robot system can pick up and return a designated object to a human. No object learning is done, meaning that even if the same object is required for another task it needs to be pointed at again. In many conditions such an explicit object model is desired.

Conclusion

To overcome inaccuracies inherent to natural pointing and inaccuracies in the recognition methods of natural pointing, several assisting devices have been proposed to accurately resolve a natural deictic pointing gesture. Some of these devices are complicated and require adaptations to the environment and to the referents. A simpler device that has been used in combination with robotics systems is the laser pointer, it requires no adaptations to the environment and works with a large range of referents. By using the laser pointer, deictic references can be resolved accurately by an automatic system while the gesture remains close to the original natural pointing gesture. For these reasons the laser pointer is also used in this research.

2.3 Object Segmentation

The first step in many object learning systems is that of segmentation, discarding information that is not part of the target. In computer-vision this is translated in defining foreground and background. The segmentation problem is linked to object learning, as an improper segmentation will logically lead to a less successful object model. As discussed in section 2.1, the object concept depends on many factors and it is not possible to create one single definition of an object useful for automatic segmentation using only visual information. This is why fully automatic segmentation methods used in computer-vision are never perfect (Boykov and Jolly, 2001). Many segmentation techniques use human knowledge because of this, these methods are described in section 2.3.1. By using a mobile robot system, it is possible to create systems that do not explicitly rely on a human teacher, such systems are described in section 2.3.2.

2.3.1 Supervised Segmentation

Manual segmentation by a human is the most accurate method of segmenting objects, but this method is very tedious and unsuitable for (semi-) autonomous behavior of the robot. Nonetheless, this method is still often used according to Ekvall et al. (2006), and there are several recent examples (Andreasson and Duckett, 2003, Gopalakrishnan et al., 2005, Lomker and Sagerer, 2002, Zickler and Veloso, 2006). Another means to ease segmentation is to use a severely simplified world, recording images against a blank background (Deinzer et al., 2000, Moreels et al., 2007, Paletta and Pinz, 2000, Schneider et al., 2005) or by manually cropping a cluttered image to contain the object (LeCun et al., 2004). In these cases no strict manual segmentation is done, but there is a need for intensive human interaction.

Other segmentation methods put less or no strain on the user and make it possible to do segmentation in more complex environments. One such technique is background subtraction: for instance in Ekvall et al. (2006), the user is asked to place the object in front of the camera and this image is subtracted from a prerecorded background. Lomker and Sagerer (2002) use a similar approach, this system consists of a dialog system and an image processor which uses a static camera. When the system encounters an unknown object it asks the user to remove it from the scene, it then subtracts the view before and after the user interaction to find the object. Instead of using only two images, a system can also use a visual stream. In Roth et al. (2006) a system is presented that segments objects based on a background model and the use of image differencing on the visual stream. This method segments moving objects well, but is very restrictive in that it excludes objects that cannot be moved by a human and more importantly, the camera view itself cannot move which limits its use on a mobile robot.

As in our approach, the systems described by Moller et al. (2005) and Toptsis et al. (2004) combine deictic gestures and computer-vision techniques to segment objects. Using their BIRON speech system a user can point to an object and utter a target color. The user needs to remain close to the object however

and objects need to have a bland color pattern. The system can then extract a basic segmentation of the object.

Various computer-vision techniques exist to segment objects from a scene using image data such as color and edge information, with only limited user interaction. Some examples of such methods are Snakes (also called active contours) which uses contour-information (Kass et al., 1988), Graph Cut (Kolmogorov and Zabih, 2002) and an extension of the latter, GrabCut (Rother et al., 2004). In all of these methods the user provides an initial segmentation, which is then enhanced.

Both GrabCut and Graph Cut use a Min-Cut/Max-Flow algorithm on boundary information for segmentation. Graph Cut represents all pixels in an image as nodes in a graph, where the strength of the edge weights on the graph are defined by a cost function defined by boundary information found in the image. A Min-Cut/Max-Flow algorithm is then used to minimize a cost function, which is defined in such a way so that a minimum of this function corresponds to an optimal segmentation.

GrabCut extends on this idea by using a Gaussian Mixture Model (GMM) to model color information as well, and it is run iteratively identify instead of once (Rother et al., 2004). GrabCut requires a rough initial guess as to the location of the foreground object, which is refined until convergence on an energy function is reached (see also section 3.1.2). The GrabCut algorithm has been used with success on a mobile robot by Moller et al. (2005). Due to its reliance on color the GrabCut method can remove parts of the foreground, if the same color is present in the background. Prakash et al. (2006) have successfully combined Snakes and GrabCut into a system called SnakeCut, which solves this problem.

Instead of solely relying on image-information, other sensors can be used for enhancing the segmentation by getting depth information about the scene. These are for instance a range finder (Nicolescu and Mataric, 2001) or stereo vision (Wersing et al., 2007). By using more information the certainty of a segmentation is increased. The advantage of stereo vision over a range finder is that there is no additional problem of combining depth and vision information.

Conclusion

Several methods for assisted object segmentation exist and the need for human help with these methods varies. By using stereo vision or a range finder the need for human assistance becomes even less. There remains a need for human interaction, because the object definition cannot be made concrete. By using a computer-vision technique as GrabCut an accurate object segmentation can be made with only limited user interaction.

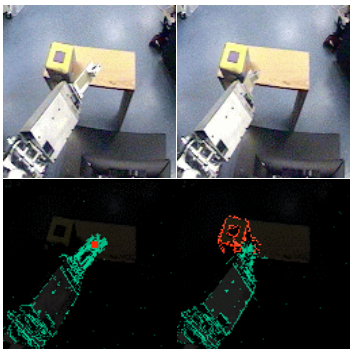
2.3.2 Unsupervised Segmentation

Instead of using human guided information to segment objects from a background, it is also possible to have a system do segmentation automatically. Several robotic systems can segment objects based on simple assumptions or by using active exploration.

Gopalakrishnan et al. (2005) present a system capable of segmenting objects automatically, based on salient color features of the object. The system contains a couple of basic assumptions on the colors present in its environment and based on these assumptions it can detect new objects. The method is limited to very specific objects, as only those with a sufficiently bland hue which does not overlap with the basic assumptions can be segmented. These are very strong assumptions, and the method only works in a limited amount of environments.

Another method that can be used for unsupervised segmentation, is active vision, also called animate vision (Ballard, 1991). Active vision involves using a sensor to see different views of an object in a purposeful manner (Dutta Roy et al., 2004). By trying to manipulate the environment, an active recognition system can find out which things are connected to each other and thus part of the same object, or it can manipulate its own sensors to view different sides of an object. Such operations are one of the main advantages of using a robotic system over a passive observance system (Pfeifer and Scheier, 1999). Besides being useful for object segmentation, active vision can be used to create more stable object models (see section 2.4).

The robot Cog uses active exploration to successfully segment objects (Fitzpatrick, 2003). The robot can prod an object to explore its boundaries and other physical properties (see figure 2.3(a)). Because of



(a) The robot Cog segments an object by prodding it, image from (Fitzpatrick, 2003)

(b) Segmenting an object by circling it, image from (Kootstra et al., 2007)

Figure 2.3: Two examples of using active vision

the use of active exploration, the set of objects that the system can handle is quite large, but it is limited to those objects that can be moved by the robot. The set of objects that can be segmented unsupervised is further limited to those objects that are placed in front of the robot by a human experimenter.

In Kootstra et al. (2007) the robot actively explores objects in a different manner: the robot keeps its camera focused on an object while circling it (as can be seen in figure 2.3(b)). Segmentation is done by looking at the field of flow, this differs between object and background. The subset of segmentable objects is larger, as there is no limitation on the movability of the object. However, the robot needs to be positioned in such a way that it can make a full circle around the object while keeping this object in its view. The system can not keep the camera pointed towards the object automatically and if non-target objects are placed near the target object the system cannot distinguish between these.

Conclusion

Several systems exist that can automatically segment an object from other objects and the background, most of these systems have severe limitations, however. A very promising method for automatic object segmentation is to use active vision. Even though object segmentation is done autonomously, examples from current research show that human assistance remains needed, as objects need to be placed near or in front of the robot.

2.4 Object Representations and Recognition

After segmentation an appropriate object model must be created to allow for recognition in later images. A distinction can be made between two types of representations, model-based representations which use the shape of an object for recognition (examples include wire-frames, spatial-occupancy representations, skeleton representations and many others) and appearance-based approaches (Dutta Roy et al., 2004). Shape based matching is often very resource intensive (Dutta Roy et al., 2004). An example of such a method Johnson and Hebert (1998), where a 3D model is made of objects. With this model, it should be possible to recognize the object from different viewpoints and be robust against occlusions and light changes. However, making and matching the 3D model is not trivial and is computationally expensive (Johnson and Hebert, 1998).

Object representations based on appearance can simply consist of an image, a set of multiple images of the object, or a collection of features extracted from one or multiple images of the object (Dutta Roy et al., 2004). When using the entire image of an object as model, objects can be recognized with techniques like normalized cross correlation (Rother et al., 2004), neural networks (Gopalakrishnan et al., 2005), convolution nets or support vector machines (LeCun et al., 2004). A disadvantage of using the training images directly as object model is that recognition time quickly increases when more objects are added

and that most of these techniques are not invariant to light conditions, viewpoint changes and occlusions, making object recognition in real-world conditions impossible.

Another way of creating object models is a parts-based model, whereby features are extracted from an object instead of taking the direct observations of an object as representation. Features such as color and texture (Lomker and Sagerer, 2002, Steil et al., 2007), or size of the object can be used (Takizawa et al., 2003). A disadvantage of such simple features is that they are not very robust and that they might not be enough to distinguish one object from another (Takizawa et al., 2003). A problem with color is that it is very sensitive to light changes, for instance (Steil et al., 2007).

One method to overcome such problems is to use features that are designed to be invariant to changes and be robust against distortions. These are methods such as Scale-Invariant Feature Transform (SIFT) (Lowe, 2004), Speeded Up Robust Features (SURF) (Bay et al., 2006), Gradient Localization and Orientation Histogram (GLOH) (Mikolajczyk and Schmid, 2005), or PCA-SIFT (Ke and Sukthankar, 2004). The latter three are all derivatives of SIFT.

These methods all work by identifying local interest points, which are then described in a feature vector. This vector is constructed in such a way that it can be matched onto other features with simple distance metrics such as euclidean distance. An interest point, described by a feature vector is called a keypoint. A more in depth description of this process can be found in section 3.2.1. Local interest points are found by locating local extrema, highly textured surfaces thus allow for more interest points to be found (Lowe, 2004). SIFT features are invariant to scale and image rotation and they are robust against affine distortions, rotations in viewpoint, noise and illumination (Lowe, 2004). A disadvantage of these methods is that not all objects are textured enough to contain such features and that the method is computationally intensive.

All derivatives from SIFT offer improvements on the original algorithm. PCA-SIFT uses Principal Component Analysis (PCA) to reduce the dimensionality of the feature vectors to improve the processing speed (Ke and Sukthankar, 2004). GLOH is optimised for better performance, but comes at a higher computational cost (Mikolajczyk and Schmid, 2005). SURF has been optimized for speed while keeping its performance on par with SIFT (Bay et al., 2006). This makes this last algorithm very interesting to use on mobile robots, where real-time performance is necessary and computational power is limited while the need for robustness is high. Experiments done by Bay et al. (2006) on 216 images of 22 objects in an art museum setting show that SURF outperforms (scoring a 82.6% recognition rate), GLOH, SIFT and PCA-SIFT (which scored, respectively: 78.3%, 78.1% and 72.3%).

As these keypoint based methods are computationally intensive, Ekvall et al. (2006) create object models both as a collection of SIFT keypoints and a Receptive Field Cooccurrence Histogram (RFCH). The histogram matching is used for object detection at a larger distance: using RFCH hypotheses of object positions are generated. The robot then actively explores these positions to try and confirm these hypotheses by extracting and matching keypoints found at this location.

As described in Dutta Roy et al. (2004), much can be gained from taking multiple views of an object during the learning and recognition stages of an object using active vision. Features of an object might be occluded by the object itself, meaning that using only one view of such an object would result in a low recognition rate (Dutta Roy et al., 2004). There is also ambiguity between objects which cannot be solved by only one view such as a *Sedan* and a *station wagon* having the same front ends, but different sides (Gremban and Ikeuchi, 1994). Much of what is called active vision within computer-vision is based on using a very static set up, having a rotating platform or rotating camera (Deinzer et al. (2000), Moreels et al. (2007), Paletta and Pinz (2000), Schneider et al. (2005) and many others). Object learning and recognition in these computer-vision systems is not done in a realistic setting (see for instance figure 2.4) and the applicability to the robot domain is thus limited. However, such research does show theoretical limits of several approaches and the advantages of each approach. What follows is a short overview of such approaches, please note that the performance of the different approaches are hard to compare, as most researchers do not use the same database of images.

Deinzer et al. (2000) have a dataset of five different but ambiguous toy-objects, which their system can recognise with an average recognition rate of 98.6%, using a whole-image matching technique in Eigenspace. The approach uses active viewpoint selection, the algorithm decides which subsequent viewpoint will increase the certainty of an object. Without using active viewpoint selection, instead offering a random viewpoint, the method achieves a recognition rate of 81.6%. The recognition rate is

high, but the objects are not placed in a real-world situation. The improvement in recognition rate that can be achieved by actively changing viewpoint is clear.

Similarly, Schneider et al. (2005) use segmented images recorded with a turning table as input for their learner/recognizer. They can recognise 100 objects with a recognition rate of 79.8% using a neural-network based approach, that is optimised using evolutionary optimization.

In Paletta and Pinz (2000) another degree of freedom is added. Whereas in other research only the object is rotating (around the vertical axis of the object), the camera angle is changed with respect to the object as well, recording not only its front but also the top of the object. By using a radial basis function on gaussian models of specific receptive fields as object model and view integration during the recognizing stage, they can recognize 16 ambiguous objects (toy cars and several toy animals) with up to 100% recognition rate. Showing again that actively changing viewpoint increases recognition rates.

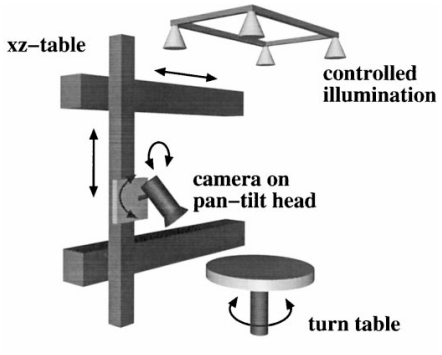


Figure 2.4: An example set up for active vision in computer-vision, image from (Paletta and Pinz, 2000). As can be seen in this image, active vision in this context is very clean, recordings are done in a simplified world where lighting conditions are highly controlled and camera movement can be predicted precisely.

Conclusion

Several means of object models exist, which differ in computational complexity and robustness against distortions. Model based object representations are robust against viewpoint changes, light changes and other distortions, but they are hard to create and require much processing power. Appearance based models are less robust and especially whole-view matching procedures remain computationally complex, but optimizations are possible making such models very attractive to use. By not storing an entire view, but instead extract features from an object the computational costs become less. Several feature extraction methods exist, of which variations of SIFT are very interesting due to their invariance against scale and rotation changes and their robustness against distortions, viewpoint changes, noise and illumination differences. Of these variations SURF seems most interesting, it is a computationally optimized version of SIFT while retaining the robust properties found in the original SIFT method. It outperforms the original SIFT method as well as several other similar methods on experiments done in a real-world setting.

By using active vision during the learning- and recognition stages the recognition rates can be greatly improved.

2.5 Related Research in Robotics

Several robot systems have been proposed to handle object learning and object recognition, having the same focus of the current research. Of these only a few use the extra possibilities that a robot platform offers over a static camera, or the option of human interaction with the robot. What follows is an overview of robot systems dealing with human interaction, object recognition using active vision or recognition in

real-world settings. It should be noted that it is difficult to compare the performance of these systems, as they all use custom datasets.

The system of Andreasson and Duckett (2003) uses a similar robot as used in this research and their system is used to recognize objects from an office environment. Their system creates object models by storing the distribution of simple features (Shi and Tomasi, 1994) over the vertical axis of the object. The system can recognize normal office environment objects (nine in total) in a real-world situation with an accuracy ranging from 63% to 100%. Due to the type of object model it cannot handle objects rotated over a different angle than the vertical axis, such as fallen objects. Because the system uses a simple descriptor, only the distribution of features over one axis is stored and no matching of these features is done, the method may not be suitable for large sets of objects. Furthermore, the system needs pre-segmented images to create object models instead of having a natural human-robot interaction.

An approach with more natural human interaction is that that of Ghidary et al. (2002). In this system an object is referred to by a human by speech and a deictic gesture. Mapping and navigation are integrated into the system as well, but the system is limited to objects that can be represented as square blocks. However, the system cannot recognize objects or update this map on its own.

The system described by Moller et al. (2005) also uses human guided segmentation, based on speech and optionally gestures to create an object model. The BIRON system is based on a similar robot as used in this research. A human user describes a target color, after which the system uses this color target and GrabCut to segment the object from the background. The system stores multiple views of an object by driving around it, but these views are only stored not processed. No recognition rates are available for this approach, however.

In contrast, the system of Kootstra et al. (2007) uses active vision in the creation of object models and the recognition of such objects. The robot drives around an object in a circle and keypoints are extracted at several positions. The method is shown to work in a complex background, with a recognition rate of up to 90% using active recognition. The robot needs to be positioned in front of the object and then follows a predefined path. It creates an object model by storing SIFT features for different viewpoints in a growing when required network, filtering out keypoints that were not stable against the change in viewpoint. The method requires for the object to be separated from the background in space and the robot needs to be placed in front of the object.

As Takizawa et al. (2003) shows, human interaction does not have to be limited to the learning stage. The system uses a human in the recognition stage as well, to confirm object positions and the system can learn from such interaction. The system uses color and size information for the object model, which is insufficient to accurately recognize the object, however Takizawa et al. (2003) claim that the user confirmation does increase the recognition rate. This was not empirically tested, however.

Not many systems are tested in real-world situations, where the objects are placed in a natural setting. Both Zickler and Veloso (2006) and Ekvall et al. (2006) describe systems that were tested in such a way. The first system uses PCA-SIFT to extract features from multiple viewpoints of an object, optionally from multiple viewpoints of the object (Zickler and Veloso, 2006). User interaction is limited to an on-screen segmentation step. Each keypoint within the segmentation is stored together with an estimation of this keypoint's position towards the object centroid, which is taken to be the centre of the segmentation. Before storing keypoints, clustering is applied to the set of keypoints to reduce the size of the object representation. In the recognition stage, each matching keypoint votes for the location of an object centroid in the image stream. These matches are clustered over time and space, to reduce the influence of noise and to accumulate evidence of object centroids. The method not only recognises objects in a scene, it also designates the position of these objects in the image. The system was tested on real-world situations, but only with two objects (a telephone and a pudding-package) in two backgrounds. In these conditions the approach works very well, recognizing the object in 92%-95% of the images. The dataset is very sparse, however, and it is not shown that these percentages are generalizable to other situations, with more objects or in different backgrounds.

Ekvall et al. (2006) use two different object models, for different recognition distances. A RFCH is stored for objects at a large recognition distance, while a model based on SIFT-features is created to confirm matches from the RFCH-method. The robot actively explores an environment by driving around and during this the RFCH object models are used to generate hypothesis of where objects might be located, the robot zooms its camera on such locations to confirm a possible match using the SIFT

model. The robot actively maps its environment and can designate recognised objects on a map. Four simple objects (a cup, a book, a rice package and zip-disks) can be found by the system with a good consistency in real-world conditions, 66%-100% recognition rate.

2.6 Conclusion

Several methods exist to create object models with robots, but there is a huge diversity in approaches. What is found in much of the robotics research is that the objects have been pre-segmented before learning by explicitly creating manual object masks or that the world is greatly simplified making object segmentation easier. It would be more natural if such a segmentation step could be integrated in the robot behavior, which is not trivial as the object concept is hard to formalize. As it can be expected that humans and robots will interact more in the future and because humans have knowledge about objects, it seems logical to integrate the segmentation in such a way as to use the human knowledge of the object.

It is hard to compare the object learning and recognition approaches within robotics, as researchers do not use the same or even comparable datasets. The size, type and number of objects differ between different research groups, as well as the testing methods. Unfortunately, it is custom within robotics research to use a low number of objects to test the validity of the approach. The number of objects that is useful for a robot to learn of course depends on the task domain, but a dataset consisting of only two object or even seven objects seems small for any domain.

Within computer-vision there has also been much attention to object segmentation and object learning. The number of objects contained in the datasets in computer-vision research is much larger. However, due to the method of recording, these datasets are often limited to one size of objects and the datasets do not represent a real-world situation. The research in computer-vision is more theoretical, it offers insights in the influence of various parameters and can show strengths and weaknesses of several methods. As the recordings are made in simplified worlds, the applicability to the robot domain is limited. Sensors are much noisier and the world is much more complex.

Human interaction in combination with object learning is limited in much of the robotics systems. The human user has to limit his or her interaction, by using a subset of natural speech or by restricting his or her gestures. Even then there is a considerable error rate of designating objects to a robot in such a way. Another disadvantage is that, by their design, some object referring methods exclude certain objects from being taught to a robot system. Using specialized devices for this task improves the recognition rates, but introduces new problems, such as the complexity of the devices or the need for a special adaptation of the environment. One way to circumvent these problems is to use a simple pointing device that still allows the user to designate objects over a large distance with ease, such as a laser pointer.

During the creation of an object model and during the recognition stage, it pays off to use active exploration. Especially in an embodied system, such as a robot, which has to operate in an uncertain world it is necessary to have robust object models and certainty in the recognition stage.

A system combining all three factors, human-robot interaction for aided segmentation of an object, active exploration of an object during the learning stage that is capable of learning a large amount of objects differing in shape, size, color and texture has not yet been proposed, and would be a useful addition to the body of knowledge of robotics. The research done in this work tries to create such a system. By making use of the knowledge of objects that a human teacher has, an object can be referred to and an initial segmentation of the object can be made for the robot. Creating such a segmentation should happen in a way that is natural to the user, while the recognition of this segmentation should not be too complex for the system. This means making use of a laser pointer, the laser pointer is a common device which is natural to use while still making it easy for an automatic system to resolve the deictic reference made with it. After this interaction the robot will create an object model by circling the object, making use of active vision to create a robust object model. The model is based on SURF keypoints, as this method has been shown to work fast on a mobile robot, while the features that are created with SURF are robust against various distortions and transformations. The method will be tested on a mobile robot, using an extensive dataset in a real-world to show how the system performs in a normal operating procedure.

Chapter 3

Methodology

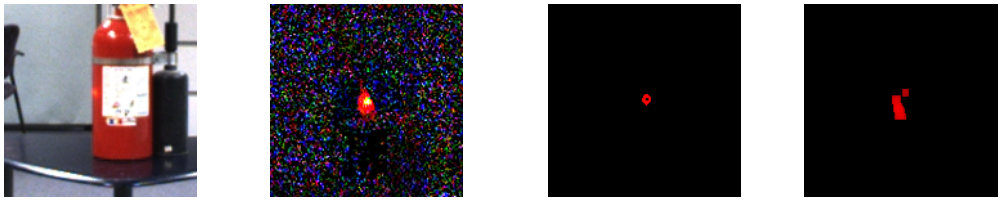
3.1 Object Segmentation

Object segmentation is done using human assistance in the system proposed in this work. The human teacher starts by aiming a laser pointer at an object and "painting" the object with the laser beam. The robot's stereo camera identifies the reflection of the laser point projected on the object and tracks its position in 3D using its stereo camera as the point moves around the object's surface. After the human turns off the laser pointer the robot uses the recorded positions of the laser reflections to segment the object from the background. Optionally a computer-vision algorithm called GrabCut (Rother et al., 2004) can be used to enhance this segmentation. These steps are described in section 3.1.1 through 3.1.2.

3.1.1 Laser Detection

The detection of the laser pointers is based on algorithms described by (Kemp et al., 2008, Kirstein and Mueller, 1998, Olsen Jr and Nielsen, 2001). In both Olsen Jr and Nielsen (2001) and Kemp et al. (2008) a brightness threshold is used to detect the presence of a laser pointer. In the case of Kemp et al. (2008) a physical filter is used to filter out all except for a narrow band of colors and the position of the detection is further refined using motion detection. Kirstein and Mueller (1998) rely solely on motion detection to estimate the position of the laser pointer, after this a matching procedure is done to remove erroneous detections in all three of these methods.

The first step to detecting the laser pointer in this research, is motion detection. Part of the teaching protocol for the human is to move the laser pointer over the object, so it can be assumed that the pointer is moving. After this a thresholding is applied on the color values in the red channel and blob detection is done to find an appropriately sized detection. What follows is a description of this in more detail.



(a) Laser pointer as seen in the image (b) Motion detected (c) Color thresholding (d) Dilation

Figure 3.1: Four steps in the procedure of detecting the laser pointer. Images (b), (c) and (d) have been enhanced to show greater detail when printed

The first step in detecting the laser pointer is the use of motion detection, this removes most of the

image pixel data save for noise, reflections and the projection of the laser pointer itself. The result of this can be seen in figure 3.1(b), note that the this image is enhanced in such a way as to show the noise as well as the laser detection. To detect the laser pointer in image I_n, the previous frame $I_{(n-1)}$ is subtracted from it, resulting in a difference image I_d:

$$I_d(i,j) = I_{(n)}(i,j) - I_{(n-1)}(i,j) \tag{3.1}$$

Step two is filtering the color information. As the laser pointer used in this research is red, all information from the green and blue color channels is discarded. The laser pointer has a high value in the red channel and therefore all pixels below a threshold of 70 (out of 255) are also discarded. The result of this operation can be seen in figure 3.1(c). Step three is a dilation, which is a morphological operation performed on the remaining pixels. A dilation is a convolution of an image with a *kernel* and can be seen as expanding regions Bradski and Kaehler (2008). In this case the kernel is a $3x3$ square mask. The dilation is necessary as the thresholding might sever part of the laser detection. If the human moves the laser pointer fast, the detection will look like a line instead of a dot. In such a case the thresholding might sever the detection meaning that it breaks up in parts which may not be large enough to count as a detection, which in most cases is solved with this dilation. This can be seen in figure 3.1(d). The last step is to filter remaining image data using a connected components algorithm[1]. This method is run to generate contiguous blobs and the largest blob that is smaller then the size constraint of 200 pixels is selected as the position of the laser point.

Clustering of Laser-point Detections

It is still possible that the detected laser reflection is incorrect, the human teacher could have made a mistake or the algorithm has detected a false positive due to noise or reflections. To filter such detections not belonging to the object, the laser pointers are clustered based on their position in 3D space. The location in 3D space is calculated for each laser detection using the stereo vision system as described in appendix A. Laser detections belonging to the object are expected to be close together in 3D space, while outliers such as reflections or accidental pointing of the background are alone and apart from other detections in 3D space. For each new laser detection the distance to each existing cluster is calculated, where the distance to a cluster is defined as the distance to the centroid of this cluster. The largest cluster is considered to be representing the object.

Objects come in different sizes and shapes, and for this reason a simple distance metric such as euclidean distance is not sufficient for clustering. For instance, for a large object the distance of new laser detections to the existing object cluster centroid can be large, while the same distance would mean that the detection does not belong to the object for a small object. Another problem with using euclidean distance is that it does not account for different shapes. For objects with a rectangular shape such as a high chair, the distance in one dimension does not mean the same as the distance in another dimension. For these two reasons the Mahalanobis distance is used. Where with a euclidean distance metric an object is always represented as a sphere shape, the Mahalanobis distance takes the correlation matrix of a cluster into account meaning that an object can be represented as an ellipsoid shape. The Mahalanobis distance metric also differs from the euclidean distance in that it is scale invariant, meaning that the same value can be used for objects of different sizes.

Clustering laser detections is done on the fly. Immediately after detecting a laser pointer reflection on a surface at location $\vec{x}$, the Mahalanobis distance d_M to the centroid μ of each existing cluster is calculated, see (3.2), with Σ being the covariance matrix.

$$d_M(\vec{x}) = \sqrt{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})} \tag{3.2}$$

The detection is added to the closest cluster, if the distance is less than 0.05. A new cluster is created if there are no clusters within this distance. Clusters with less than three elements contain not enough information to calculate a covariance matrix, so for these clusters a standard covariance matrix is used 3.3. An example of laser detection clustering can be seen in image 3.2. In this image some laser pointers were incorrectly detected in the background and others were given a wrong position in 3D by the stereo vision

[1] `http://opencv.willowgarage.com/wiki/cvBlobsLib#Documentation` using the version released on 2008-08-24

algorithm. As can be seen in the image, this results in three clusters. The middle cluster represents the object, while the cluster in front of it (on the x-dimension) contains incorrect depth measures from stereo matching and the cluster after it are detections in the background. Note that clustering is performed in 3D, but that in this image the y-dimension is flattened for display purposes. Without clustering the object segmentation would be incorrect on the x-dimension, the object segmentation would be much bigger than the object actually is. The same holds for incorrect segmentation in the other two dimensions.

$$\begin{pmatrix} .1 & 0 & 0 \\ 0 & .1 & 0 \\ 0 & 0 & .1 \end{pmatrix} \tag{3.3}$$

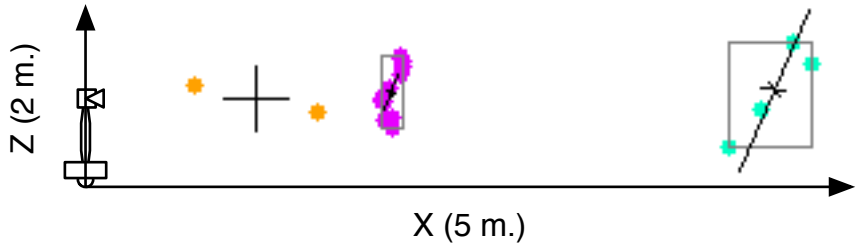


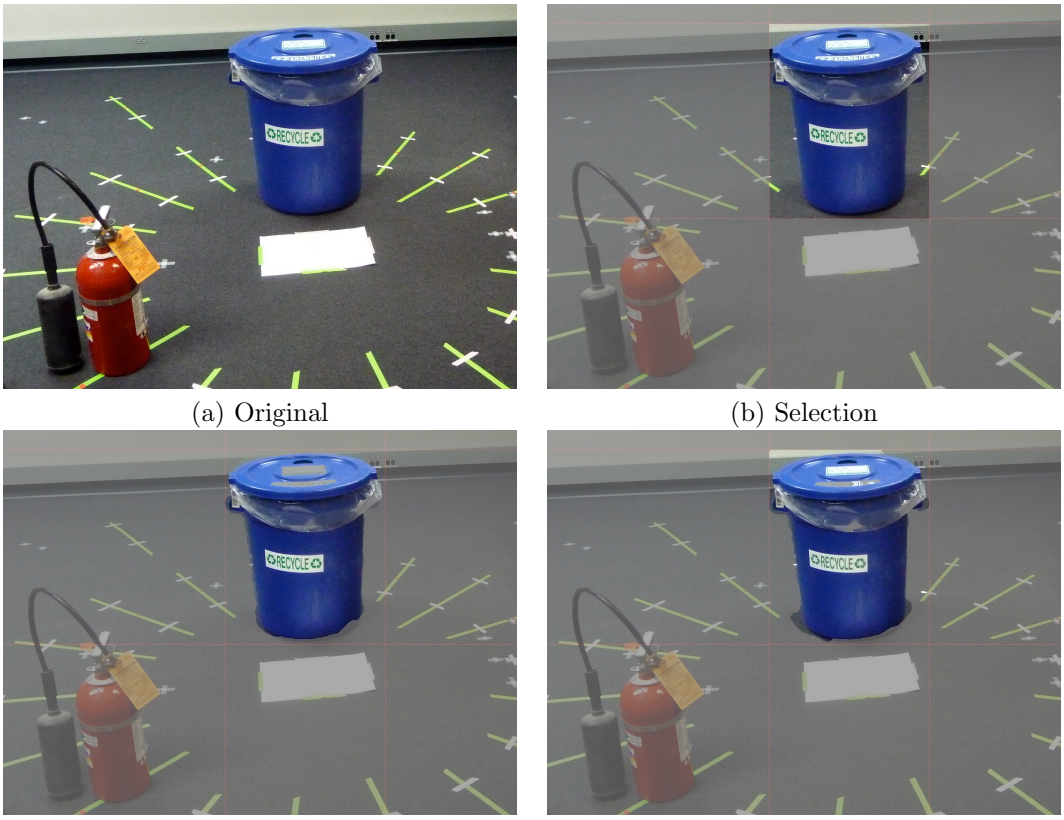
Figure 3.2: A mapping of three clusters of detected laser points from 3D space to 2D. The human teacher and the robot are located at the origin, the object is described by the middle cluster, and the other two clusters contain incorrect detections. The z-axis describes two meters, the x-axis five meters. The crosses show the covariance matrix

If at least 30 frames have passed without a laser detection, the laser pointing period has passed. Instead of needing a separate stopping condition, such as a word or other signal, the object referral period thus ends in a natural way. The largest cluster is considered to be describing the object. The bounding box around this largest cluster is used as the preliminary object segmentation in the image plane. This segmentation contains the object, but in most cases also a large amount of background pixels. This is because not every object can be described using a box. To enhance this segmentation by discarding the background pixels, a computer-vision technique called GrabCut (Rother et al., 2004) can be applied.

3.1.2 GrabCut

GrabCut is a segmentation technique which requires only limited information to perform a good segmentation (Rother et al., 2004). The algorithm combines color and boundary information and is an extension of a technique called graph cut (Boykov and Jolly, 2001). GrabCut differs in that it uses color information, does iterative optimization and that it can handle incomplete labelling (Rother et al., 2004). The standard GrabCut algorithm includes a border matting procedure, which produces nicer looking segmentations by applying a gradient in alpha around the border of segmentation. As GrabCut was used to acquire a hard segmentation between object and background this matting technique was not used. An example of an initial segmentation which is enhanced can be seen in figure 3.3.

The core of GrabCut is the graph cut segmentation technique as proposed by Boykov and Jolly (2001). This technique represents pixels in an image as nodes in a graph and then uses a Min-Cut/Max-Flow algorithm to divide the graph into a fore- and background (Boykov and Jolly, 2001). The weights in the graph are defined by an energy function, which depends on boundary and region information. As Boykov and Jolly (2001) use an intensity histogram to model region information, their implementation does not utilise color information. Boykov and Jolly (2001) chose to make use of a Gaussian Mixture



(a) Original (b) Selection

(c) After two iterations (d) Final segmentation

Figure 3.3: As can be seen here, the GrabCut segmentation technique is able to segment a basic object using only a bounding box (b) as guidance. The bounding box is used as an initial segmentation, which is enhanced over several iterations. Oversegmentation can be seen in (c), as part of the top of the object is seen as background. After more iterations this is no longer the case, as can be seen in (d).

Model (GMM) instead of histograms, to include color information. There are two GMMs, one for the foreground and one for the background each with $K = 5$ Gaussian-components.

In the current approach, the initial segmentation is taken as the bounding box of the largest cluster of detected laser reflections. All pixels not in this bounding box are considered background, all others are considered unknown. The initial labelling α_n is taking $\alpha_n = 0$ for the background and $\alpha_n = 1$ for the unknown pixels. Both fore- and background are modeled with a full-covariance Gaussian mixture with $K = 5$ components as suggested by Rother et al. (2004). The following explanation is based on Rother et al. (2004). The initial GMM models are created using the Expectation Maximization (EM) clustering algorithm. Each pixel is assigned a unique GMM component k_n (with $k_n \in (1, \dots K)$), from either the background or the foreground model depending on the labelling of the pixel ($a_n = 0$ or 1). Now a "Gibbs" energy function E can be defined as in (3.5), this energy function is defined in such a way that minimization will give an optimal segmentation. This function depends on the segmentation $\underline{\alpha}$ of data $\vec{z}$, with parameter model $\underline{\theta}$ defined (3.4) and the corresponding GMM component variables $\vec{k}$. The parameter model $\underline{\theta}$ consists of the weights (π), means (μ) and covariances (Σ).

$$\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0, 1, k = 1 \cdots K\} \tag{3.4}$$

$$E(\underline{\alpha}, \vec{k}, \underline{\theta}, \vec{z}) = U(\underline{\alpha}, \vec{k}, \underline{\theta}, \vec{z}) + V(\underline{\alpha}, \vec{z}) \tag{3.5}$$

The energy function (3.5) has two terms, a data term U and a smoothness term V. The data term evaluates the fit of $\underline{\alpha}$ on z, while the smoothness term encourages coherence in regions of a similar color (Rother et al., 2004). The data term is defined (3.6), with D defined (3.7).

$$U(\underline{\alpha}, \vec{k}, \underline{\theta}, \vec{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n) \tag{3.6}$$

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -log p(z_n|\alpha_n, k_n, \underline{\theta}) - log\pi(\alpha_n, k_n) \tag{3.7}$$

In this term the $p(\cdot)$ is a Gaussian probability distribution and $\pi(\cdot)$ are mixture weighting coefficients. This means that the function can be rewritten (up to a constant) to (3.8).

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -log\pi(\alpha_n, k_n) + \frac{1}{2} log|\Sigma(\alpha_n, k_n)| + \frac{1}{2} \left[z_n - \mu(\alpha_n, k_n)\right]^T \Sigma(\alpha_n, k_n)^{-1} \left[z_n - \mu(\alpha_n, k_n)\right] \tag{3.8}$$

Where π are the weights, μ are the means and Σ are the covariances of the Gaussian components k for the fore- and background distributions. The smoothness term V is described using (3.10), with $\gamma = 50$ and β defined (3.9) and C the set of neighboring pixels with 8-way connectivity.

$$\beta = \frac{w \times h}{4} \times \left(\sum_{(i,j) \in I} \left(r(i,j)^2 + g(i,j)^2 + b(i,j)^2\right) - \sum_{(i,j) \in I}{}^2 r(i,j) - \sum_{(i,j) \in I}{}^2 g(i,j) - \sum_{(i,j) \in I}{}^2 b(i,j) \right)^{-1} \tag{3.9}$$

$$V(\underline{\alpha}, z) = \gamma \sum_{(m,n) \in C} [\alpha_n \neq \alpha_m] \exp^{-\beta ||(z_m - z_n)||^2} \tag{3.10}$$

After the initial segmentation has been made with the bounding box around the laser detections and the first GMMs were created using EM, iterative energy minimisation is done. This consists of three steps, minimizing the total energy E with respect to k, $\underline{\theta}$ and $\underline{\alpha}$ in turn. For each pixel n in the unknown set, a k is assigned (step 1). Then GMM parameters $\underline{\theta}$ are updated by estimating them from the current segmentation (step 2). For each component k the mean and covariance matrix are estimated to be the sample mean and covariance of the pixels in $F(k) = \{z_n : k_n = k \text{ and } \alpha = 1\}$, the weights are estimated to be $\pi(\alpha, k) = |F(k)| / \sum kF(k)|$. Finally a segmentation estimation is done by performing a global optimization using MinCut to find the optimal $\underline{\alpha}$ for 3.5 (step 3). The algorithm is guaranteed to converge to a minimum of E and in the paper by Rother et al. (2004) these three steps are indeed

run until E converges. In the current implementation the three steps were run only six times to limit processing time, which worked well enough for this approach. The labelling z_n is taken as the final segmentation.

3.2 Object Learning

After the user has designated an object and it has been segmented in the image plane, the object model can be made. The first step is calculating a bounding box around the object in 3D based on the detected laser pointers (section 3.2.2). The robot then drives around the object in a circle, stopping at several locations to create an object model based on several "views" of the object. The robot uses Speeded Up Robust Features (SURF) (Bay et al., 2006) (see section 3.2.1) to create this model and uses active vision (Kootstra et al., 2007) and the bounding box to filter out unwanted keypoints (section 3.2.2).

3.2.1 SURF feature extraction

As described in Bay et al. (2006) the SURF algorithm creates descriptors in the image domain that are robust to noise, detection errors and geometric and photometric deformations. The SURF algorithm consists of two steps, first 'interest points' are selected, then the neighborhood of these points are represented with a feature vector. This resulting combination of a location with a descriptor is then called a keypoint. Repeatability in creating keypoints is very important here: the algorithm should pick the interest points consistently under different viewing conditions. The same holds for robustness, the feature vector describing an interest point in one image should not differ too much from an feature vector describing the same point in a different image. Finally, speed and computational costs are also important and these depend heavily on the dimension of the descriptor and the complexity of the algorithm to create it. The algorithm is optimized for speech by limiting the size of the feature vector to 64 elements and by reducing the level of invariance to image scaling and rotation (not considering skew, anisotropic scaling and perspective changes). The authors argue that the additional complexity of trying to account for other deformations does not pay off and that the general robustness of the algorithm is enough to deal with such second order effects. An example of an image with SURF features extracted can be found in figure 3.4.

The SURF algorithm uses a Fast Hessian Detector (Bay et al., 2006)to calculate interest points and then assigns a reproducible orientation to these interest points. The descriptor then describes the distribution of Haar-wavelet (Haar, 1910) responses in a region centered around the interest point, with the orientation of the region equal to the orientation of the interest point. The following explanation is based on the article of Bay et al. (2006).

The Fast Hessian interest point detector uses the determinant of the Hessian for selecting a location and size of the interest point instead of using different measures for each (as done in (Mikolajczyk and Schmid, 2005)). For each point $\vec{x}$ in an image I, the Hessian matrix is defined as:

$$\mathcal{H}(\vec{x},\sigma) = \begin{bmatrix} L_{xx}(\vec{x},\sigma) & L_{xy}(\vec{x},\sigma) \\ L_{xy}(\vec{x},\sigma) & L_{yy}(\vec{x},\sigma) \end{bmatrix} \tag{3.11}$$

Where L_{xx} is the convolution of the Gaussian second order derivative $\frac{\delta^2}{\delta x^2} g(\sigma)$ with the image I in point $\vec{x}$ (similarly for $L_{xy}(\vec{x},\sigma)$ and $L_{yy}(\vec{x},\sigma)$). Bay et al. (2006) use box filters (see figure 3.6) to approximate these convolutions L, which they show are equally effective as the normal discretized and cropped Gaussians. The approximations are noted as D_{xx}, D_{xy} and D_{yy} respectively. The determinant of the Hessian matrix $det(\mathcal{H}_{approx})$ then becomes:

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9 \times D_{xy})^2 \tag{3.12}$$

The weights of D_{xy} is balanced on the recommendations in Bay et al. (2006). The lowest spatial resolution is a 9x9 box filter (corresponding to a Gaussian derivative with $\sigma = 1.2$). This resolution is iteratively enlarged to larger box filters (i.e. 15x15, 21x21, 27x27, etc.) to find all interest points. Each box filter corresponds to a different σ for the Gaussian derivative, which in turn corresponds to a scale

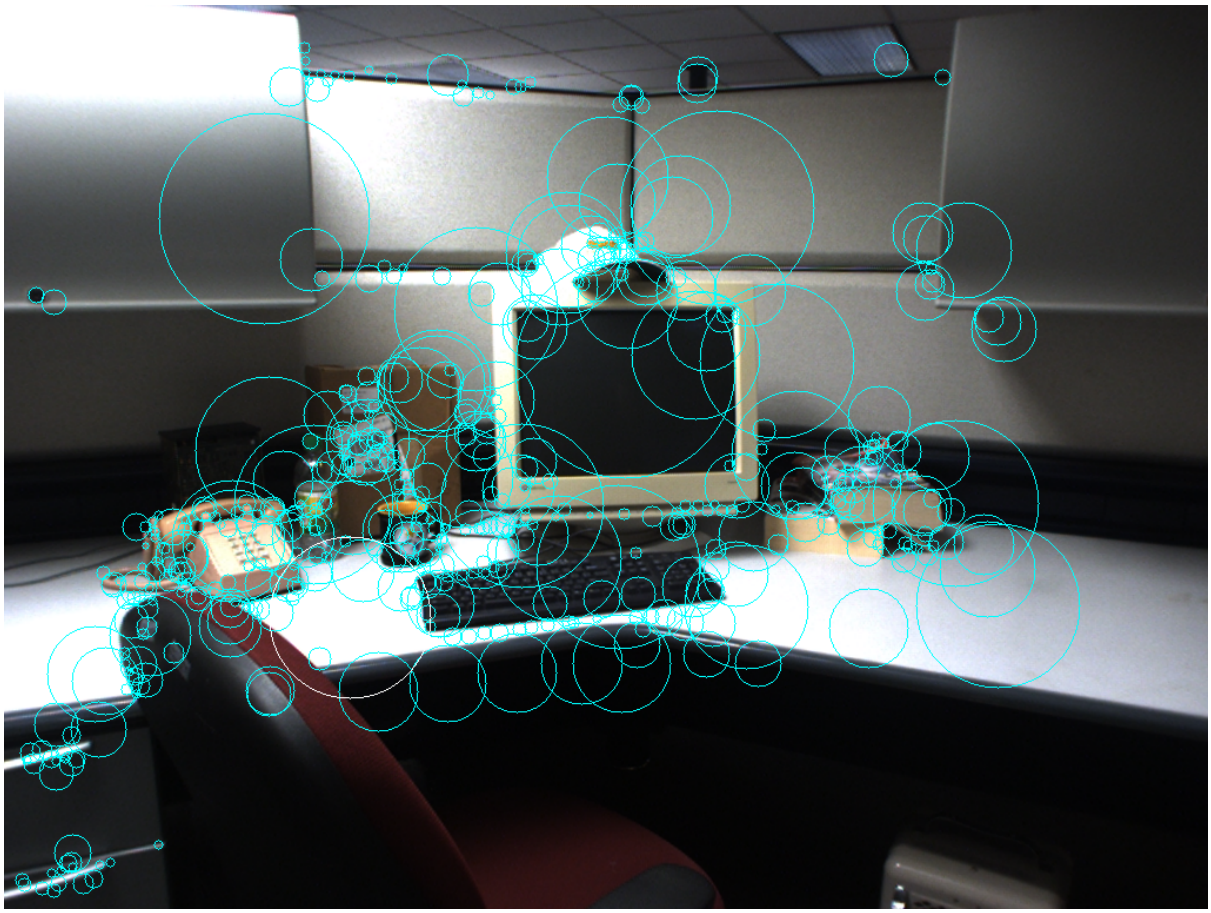


Figure 3.4: *Scene 6, trial 1* with extracted SURF features. Each circle represents a SURF feature and the size of the circles corresponds with the size of this feature.

factor s. As the ratios of the filter layout remain constant after scaling, the approximated Gaussian derivatives scale accordingly. The 27×27 filter corresponds to $\sigma = 3 \times 1.2 = 3.6 = s$.

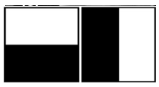


Figure 3.5: The Haar wavelets that are used in SURF to calculate the orientation of the interest points. Image from Bay et al. (2006)

The orientation of the interest points is calculated by using the Haar-wavelet responses (see figure 3.5) in x and y direction, in a circular neighborhood of with a radius $6s$ around the interest point (where s is the scale of the current interest point). The wavelet responses are calculated at the same scale s. After calculating the wavelet responses and weighing them with a Gaussian with $\sigma = 2.5s$, they are represented as vectors in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of $\frac{\pi}{3}$. The horizontal and vertical responses within this window are summed and the two summed responses then yield a new vector, the longest such vector lends its orientation to the interest point. The spatial size of this sliding window is set at 4, as chosen by Bay et al. (2006).

The actual descriptor is extracted out of a square region with size $20s$, centered around the interest

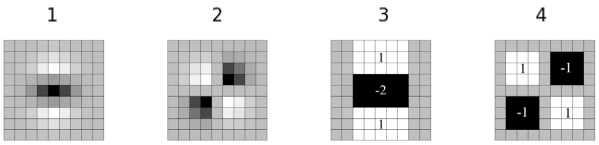


Figure 3.6: The discretized and cropped Gaussian second order partial derivatives (one and two) and the approximations thereof using box filters (three and four), grey regions are equal to zero. Image from Bay et al. (2006)

point with an orientation as calculated in the previous step. The region is split up regularly into smaller 4 x 4 square sub-regions and for each sub region a few simple features are calculated. Each subregion has a four-dimensional descriptor vector $\vec{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, with d_x and d_y being wavelet responses. The wavelet responses are calculated at 5x5 regularly spaced intervals in a horizontal direction (d_x) and vertical direction (d_y) within the subregion, where vertical and horizontal are defined with respect to the orientation of the region. The responses d_x and d_y are weighted with a Gaussian ($\sigma = 3.3s$) centered at the interest point, to account for geometric deformations and localisation errors. The wavelet responses are invariant to illumination biasses and invariance to contrast is achieved by turning the descriptor into a unit vector.

The extracted features (the feature vector, in combination with its scale and other qualities) are called "keypoints" and one can be matched to an other by using a distance metric on the feature vector. To match a current keypoint k to keypoints in a database K the nearest neighbor ratio matching strategy as described by Bay et al. (2006) was used. For keypoint k the nearest neighbor k_n and the second-nearest neighbor $k_{(n-1)}$ are found in the stored database of keypoints, using euclidean distance as a distance metric for "nearest". The distance metric then becomes:

$$D_{ratio}(k, K) = \frac{||k_n \in K - k||}{||k_{(n-1)} \in K - k||)} \tag{3.13}$$

If the distance is less than 0.7 a match is said to have occurred. Such a distance measure is preferred to using a global threshold as some descriptors are much more discriminative than others (Lowe, 2004).

3.2.2 Filtering keypoints

As the robot drives around the object, its current position is stored in the trajectory S and this position is evaluated for being a suitable place of recording a view of the object and updating the object model. The current position is considered suitable if the camera image is no longer moving. To create a reliable object model it is necessary for the camera image to be stable, as an image with much motion blur results in less extracted keypoints and thus a less reliable object model. First the robot needs to have stopped moving. The robot is considered to have stopped after it has been standing still for over two seconds, e.g. not moving more than one centimeter. Then the state of the pan tilt unit is read.

The camera image from each suitable position is stored as a view and SURF-keypoints are extracted from this image. Each keypoint's position in 3D is then calculated, based on the current world model of the robot (step 1). In the next step (2) the keypoints are filtered using a segmentation of the object in the world model based on the human teachers laser pointing session. Finally, active vision is used to discard keypoints that are "unstable" (step 3). This results in the view based object model as described in 3.2.3.

Step 1: Rewriting keypoint positions to the absolute world model

The stereo vision algorithm returns 3D positions per pixel in the camera image relative to the robot. It accounts for the position of the camera relative to the robot origin and the current pan and tilt of the camera as described in chapter A. The calculations do not account for robot movement and as such it would be difficult to map points from the pointcloud from one position of the robot to another. For this

reason an absolute world model is necessary. The origin of this world model is the first position of the robot, the position where the human teacher initiates the object learning sequence.

As the robot moves it uses its odometry to transform the points in the pointcloud to the absolute world model. First a rotation matrix is applied to the pointcloud with the robot's current orientation $\gamma(t)$ with respect to the origin in the world model. For each point $\vec{X} = (x, y, z)$ the rotated point $\vec{X}' = (x_r, y_r, z_r)$ is calculated as follows:

$$\begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\gamma(t) & 0 & \sin\gamma(t) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\gamma(t) & 0 & \cos\gamma(t) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.14}$$

The pointcloud is then translated to account for the robot movement $T_x(t)$ and $T_y(t)$. The final transformed point $\vec{X}'' = (x', y', z')$ is calculated as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x(t) \\ 0 & 1 & 0 & T_y(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} \tag{3.15}$$

Potentially, the error in position estimation of these points can be very large. This error is a combination of the error of the stereo matching algorithm, the error of the pan-tilt unit (which has a limited angular resolution, but the orientation it reports is not limited to this resolution) and the error of the robot odometry. However, in the current set-up the total error seems to be not too large (see 5.3).

Step 2: Bounding Box

Keypoints not belonging to the object should be discarded when creating an object model. To discard such keypoints a bounding box in 3D space is calculated that contains the object. The dimensions and position of the box are based on the detected laser pointers in the cluster of laser detections L that describes this object. As the front of an object obscures its back, the object depth (which is in the x-dimension) needs to be estimated and for many objects the width of the object is a good estimation for its depth. The 3D bounding box is defined by the two points $\vec{P}_{min}$ and $\vec{P}_{max}$. $\vec{P}_{min}$ is simply defined by the minimal x, y and z over all points in L. $\vec{P}_{max}$ is defined by the maximal y and z, the depth is estimated as being the minimal x plus the width of the bounding box ($\vec{P}_{max}(x) = \vec{P}_{min}(x) + (\vec{P}_{max}(y) - \vec{P}_{min}(y))$). Besides discarding keypoints not belonging to the object, this bounding box is also used to calculate the object centroid. The robot uses the object centroid to track the object with the camera as it drives around the object, using the pan-tilt unit. The object centroid $\vec{P}_c$ is simply defined as:

$$\vec{P}_c = \frac{1}{2}(\vec{P}_{max} - \vec{P}_{min}) \tag{3.16}$$

The bounding box is enlarged with 10%, to account for errors in the stereo vision, odometry and the affine transformations that are necessary to account for pan and tilt of the camera.

Step 3: Active Vision

The active vision filtering technique uses the fact that the robot explores the object, to see how robust keypoints are against changes in viewpoint. Keypoints that are robust are considered "stable", all other keypoints are discarded. As unstable keypoints are not robust against changes in viewpoint, such keypoints would only be useful in recognising the object from a specific viewpoint. Discarding such keypoints will thus not result in a severe degradation in recognition accuracy, but by discarding the keypoints the object model will be more sparse meaning that the recognition time will decrease. On top of this, some keypoints not belonging to the object may have been kept incorrectly by the bounding box filtering method due to stereo matching errors. These keypoints may also be discarded using active vision, as it is unlikely that the stereo matching algorithm makes the same error in a different viewpoint. The method used here is based on Kootstra et al. (2007), where it is shown that creating an object model

using active exploration will actually significantly improve the recognition accuracy over an object model not using this filtering technique.

For each view n a set of keypoints V_n is extracted, which is correlated with the set of keypoints of the previous view $V_{(n-1)}$ and with $V_{(n+1)}$. For each keypoint in V_n the closest neighbor in descriptor space is found in the sets $V_{(n-1)}$ and $V_{(n+1)}$ using an euclidean distance, the keypoint is discarded if the minimal distance does not fall within a boundary. This boundary is 0.6 a value based on experiments with SURF-keypoints done by Bay et al. (2006). See also figure 3.7.

The current approach differs in two ways from the approach by Kootstra et al. (2007). The first difference is that matches are not only found in the set of keypoints $V_{(n-1)}$, but that the set of keypoints $V_{(n+1)}$ from the next view is considered as well. Implementation wise this means that three views need to be stored in memory (V_m, $V_{(m-1)}$ and $V_{(m-2)}$) before being able to filter keypoints (from view $V_{(m-1)}$), but it allows for a larger difference in viewpoint change. If a keypoint is present in $V_{(n+1)}$ (the distance to the nearest neighbour in the set is < 0.6), but not in $V_{(n-1)}$ it would have been discarded in the original approach, but it will be stored in the adapted version used here. Secondly, in Kootstra et al. (2007) filtering was also done based on the distance between the matched keypoints in image space between two subsequent views. Keypoints were discarded if their location in view n differed too much from their location in $n-1$. This extra filtering was done to discard keypoints belonging to the background. Changing viewpoint while keeping the camera focused on an object will mean that pixels of the background will have a larger displacement then the pixels on the object. In the current approach such filtering is already done in step two, by the bounding box method.

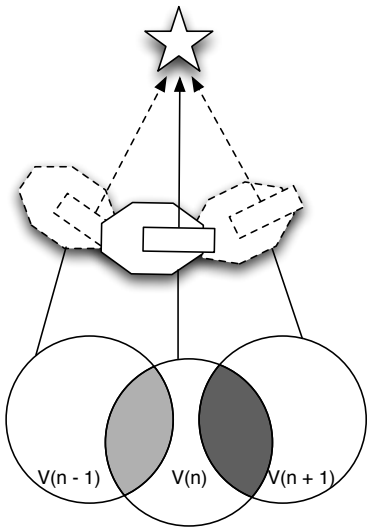


Figure 3.7: In this image, three subsequent stopping positions of the robot are seen. From each viewpoint n a set of keypoints is extracted V_n and only those keypoints are kept that can be matched with keypoints in $V_{(n-1)} \cap V_{(n+1)}$ (the light- and dark-gray areas), where a match is defined as a euclidean distance < 0.6 in keypoint space. In Kootstra et al. (2007) keypoints were only matched with $V_{(n-1)}$, the light-gray area in this figure

3.2.3 View based object model

After extracting keypoints and filtering them as described in the previous sections (step two and three) the object model is stored. The model is identified with an object identifier ID. This object model consists of a set of views Ω_{ID}. Each view $\omega \in \Omega_{ID}$ consists of the camera image taken from a certain viewpoint and the filtered keypoints belonging to that image, each keypoint tagged with its position in the absolute coordinate system. An object can be referred to with the object ID, or by referring to the set of views Ω_{ID}. An individual view ω of a certain object ID are referred to as an object-view pair

$(\omega \in \Omega_{ID} :< ID, \omega >)$.

3.3 Object Recognition

The recognition method is based on the activation model of Kootstra et al. (2007). As described in 3.2.3, the learned model for a given object ID is defined as a set of views Ω, where each view ω is a set of keypoints. The set $\mathcal{M}$ contains all keypoints from all objects and views.

To recognize objects in a new image I, a set of keypoints $\mathcal{O}_I$ is extracted from this image. For each keypoint $p_i \in \mathcal{O}_\mathcal{I}$ the nearest neighbor $k \in \mathcal{M}$ is found, where nearest is defined using euclidean distance. The activation for keypoint p_i then becomes:

$$a_i = exp(-\min_{k \in \mathcal{M}}(||p_i - k||)) \tag{3.17}$$

The total activation for an object-view pair $\langle ID, \omega \rangle$ then becomes:

$$A(\langle ID, \omega \rangle | \mathcal{M}, \mathcal{O}_I) = \frac{\sum_{p_i \in (\omega \cap \mathcal{O}_I)} a_i}{\sqrt{|\omega|}} \tag{3.18}$$

Where $|\omega|$ is the number of keypoints in this view. This normalization step is done so that fewer matched observations are needed for objects that have relatively few keypoints. This value is computed for all views of all objects and the activation level results are ordered such that the higher total activation level reflects a higher certainty that the object as seen from the specific view is in the image.

The total activation for an object ID is defined as:

$$A(ID | \mathcal{M}, \mathcal{O}_I) = \sum_{\omega \in \Omega_{ID}} A(\langle ID, \omega \rangle | \mathcal{M}, \mathcal{O}_I) \tag{3.19}$$

Optionally it is possible to filter the set of keypoints $\mathcal{O}_I$ before calculating the activation for all objects. This filtering is based on Lowe (2004). For each keypoint $p_i \in \mathcal{O}_\mathcal{I}$ the distance to the nearest keypoint n_1 in $\mathcal{M}$ is calculated. The second nearest keypoint n_2 is found from $\mathcal{M}$ as well, preferably from a different object ID. Then the ratio-distance as defined in 3.13 is calculated. If this distance is larger than 0.8 the keypoint is discarded, a value based on Lowe (2004).

Chapter 4

Experimentation

4.1 Hardware

4.1.1 Robot

The robot used in this work was a MobileRobots Peoplebot[1], as seen in figure 4.1(a), it was equipped with an on-board 1.6GHz Pentium-M single-board computer running Linux. Additionally, a Thinkpad X61 laptop with a 2.4GHz Intel Core 2 Duo processor was added to the robot to provide additional processing power.

4.1.2 Sensors

In the standard edition the robot is equipped with 24 range-finding sonars, two infra red sensors pointed upwards to detect when it might bump into objects that cannot be seen by sonar and 500 tick wheel rotation encoders for odometry estimation on each wheel. From these only the wheel rotation encoders are used in the current set-up. Additionally, the robot was also equipped with a PointGrey Research Bumblebee2 stereo-camera[2] mounted on a DirectedPerceptions pan-tilt unit type PTU-46-17.5[3].

Due to limitations in the underlying Advanced Robotics Interface for Applications (ARIA) framework, the state of the pan-tilt unit cannot be read from the unit directly. Instead of reporting the current pan and tilt angles of the unit, the ARIA framework reports the most recently set values. It is therefore impossible to read from the unit whether it is moving. The state has to be estimated from the visual stream. To check for movement of the PTU, the current frame is subtracted from the previous frame. When the summed difference is below a threshold, which depends on the resolution, the camera is considered to have stopped moving. This threshold is 4% of $(w \times h)$. The threshold accounts for noise in the sensor.

Stereo Camera

The camera is capable of extracting two color images with resolutions up to 1032x776 at a maximal speed of 20 FPS, as noted in the products' datasheet[4]. The camera can be seen in figure 4.1(b).The camera's baseline is 12 centimeters, it has a horizontal field of view of approximately 66° and a vertical field of view of approximately 52°. The camera is capable of automatically setting white balancing, gain and shutter speed based on the image of the right-camera, these options can also be set manually. For the research done in this thesis, the settings were left at auto. Besides these three options the camera does not pre-process the image, it feeds the raw images over the Firewire bus as defined by the IEEE-1394a standard. These images are interlaced and Bayer tiling remains intact, therefore the images need more processing before they can be used by vision algorithms (like stereo matching). These steps are carried

[1] http://www.activrobots.com/ROBOTS/peoplebot.html
[2] http://www.ptgrey.com/products/bumblebee2/index.asp
[3] http://www.dperception.com/products_family_ptu-d46-17.html
[4] http://www.ptgrey.com/products/bumblebee2/bumblebee2_xb3_datasheet.pdf

(a) The Peoplebot robot, the additional laptop can be seen mounted behind the SICK laser scanner

(b) Bumblebee 2 stereo camera which can be seen mounted on the robot in (a)

Figure 4.1: Hardware used in the experiments

out by the DC1394 library as described in appendix A, on page 71. All other image related operations were done using OpenCV[5].

Odometry

Estimation of the robot's position is based on the odometry method implemented in the ARIA software of Mobile Robots[6]. ARIA solely uses the wheel rotation encoders to estimate the position of the robot.

4.1.3 Laser Pointer

The laser-pointer used for the experiments is a red, class IIIa laser device with wavelengths 630-680nm. This is an original, unmodified off the shelf laser-pointing device. An image can be found in figure 4.2.

Figure 4.2: The laser pointer

[5] An open source computer-vision library of Intel, see `http://sourceforge.net/projects/opencvlibrary`
[6] `http://www.activrobots.com/SOFTWARE/aria.html`

	Resolution	Demosaicing	Rectification
Segmentation set:	320x240	Nearest Neighbor	During Recording
Other datasets:	1024x768	HQLinear	During playback

Table 4.1: Settings for recording the datasets

	Stereo mask	Edge mask	Subpixel interpolation
Segmentation set:	7	7	On
Other datasets:	13	7	On

Table 4.2: Settings for replaying the datasets

4.2 Datasets

To test the performance of the system four datasets were made. An object segmentation set, an object learning set, a real-world validation set and an accuracy-measurement set. All of these sets were recorded using the Bumblebee camera mounted on the Peoplebot. The sets were not created in the same time period and therefore differ in recording method, these differences concern the rectification method, the demosaicing method and the resolution of the camera. For an explanation of the rectification and demosaicing method please refer to appendix A. The differences are noted in the description of the datasets. The robot was carefully positioned and orientated before recording of each dataset. Stickers were added to the robot as markers to keep this consistent. With the use of a robot template (see figure B.1) the robot was carefully positioned on predefined ground-markers by aligning the markers on the template with the markers on the robot.

The environment greatly influences the performance of vision based systems, especially the lighting conditions in such environments. Colors look different under different lights, the human eye auto-corrects for this. The Bumblebee camera partially mimics this auto-correction by having automatic white balancing, automatic shutter speed and autogain. The downside of such automatic techniques, is that they may lead to different recordings in the same environment as the camera may have chosen different values for these settings. The Bumblebee camera also offers the possibility to manually set these values, but as the robot should be able to drive around in natural environments without a human having to manually set these values, this option was not used to keep the datasets more realistic. If the system is to be used in real-world conditions the values can not be tweaked either. Due to time constraints only one environment, the robot lab, was used for all datasets (with the exception of the validation set, where an additional environment was added). The robot lab offers various lighting conditions in itself as the lighting differs for different robot and object positions. The robot lab environment is comparable to a normal office environment.

As the robot lab has high intensity lighting, there is some over saturation of the Charged-Coupled Device (CCD) sensors present in the stereo vision camera, in some of the recordings. These are areas that reflect light with such a high intensity that the CCD sensors that capture this light all report their maximum value. In such a case a difference in color that is present in the environment can not be distinguished in this area. This oversaturation is unfortunate, but unavoidable in the current set-up.

The choice of background and objects also influences the performance of stereo vision, as well as GrabCut. Small and texturally sparse objects are to hard to distinguish using stereo vision. To test the influence of these factors small test-sets were created to find out the correct stereo vision settings working under a range of situations and environments. Variations were done over the distance of the object to the camera, the type and size of objects and different backgrounds. The stereo settings were chosen in such a way as to work well in different environments and for a variety of objects. They depend on the resolution and can be found in table 4.2.

The objects used in the datasets vary over size and texture to minimize the bias of the system to work only for a slight subset of objects. Objects in the set can be as large as chairs or as small as a cup. All sets containing an object-designation were created with one and the same person pointing the laser

pointer, but research by Kemp et al. (2008) suggests different people perform equally well on designating an object this way.

Movement in the datasets is limited to the laser pointer. For two other datasets, the accuracy- and object-learning sets, the robot itself is moving.

4.2.1 Object Segmentation Set

This dataset contains several objects, recorded under various conditions. It was made to test the accuracy of the segmentation techniques (see section 4.3.1). The object segmentation dataset consists of seven objects, recorded in three different positions from the robot, in two different poses and with two different backgrounds. In this set the object was pointed out with the laser pointer for thirty seconds. These objects are all from a normal office-environment: a red chair, a table, a dust bin, a cabinet, a box, a monitor and a 'lazy chair'. Images of these objects can be found in figure 4.3.

(1) Red chair (2) Table (3) Dust Bin (4) Cabinet (5) Box (6) Monitor (7) 'Lazy Chair'

Figure 4.3: Objects in the segmentation set.

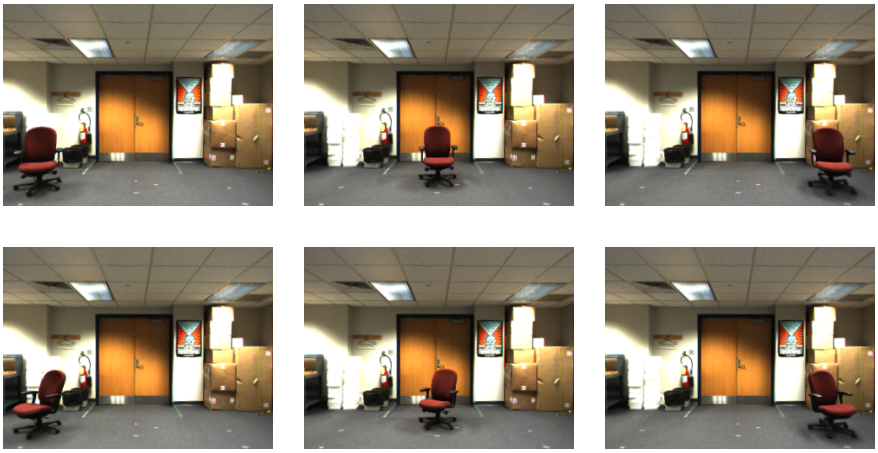


Figure 4.4: The different positions and poses of an object (red chair) as seen from the robot point of view

The experimenter was located next to the robot and waited for 15 seconds before starting the 30 second laser pointing session. Each object was located at three different positions placed in an arch at a distance of three meters from the robot as can be seen in figure 4.6. The different poses were created by rotating the object a small number of degrees as can be seen in figure 4.4. The two different backgrounds are defined as an 'empty' background and a 'complex' background as seen in figure 4.5. The objects chosen for this set are all medium sized objects from an office environment scene. The distance of three meters was chosen as it is the closest distance for which each object could be seen in full in the camera image, when the camera is in its rest-state. The rest-state is the default position of the camera, when it is not panned or tilted, the camera is in this state when the robot is turned on. Both the top and

Figure 4.5: (a) Empty background (b) Complex background

the bottom of each object can be seen in all recordings without panning or tilting the camera. A larger distance would increase the errors from the stereo matching algorithm due to the fact that the disparity between left- and right camera becomes less. Also, three meters is a distance over which humans can reliably indicate an object (Kemp et al., 2008). All objects are free of contact with other objects and the background, with a margin of at least 70 centimeter. This is justifiable because the robot will need a clear path around the objects for active vision.

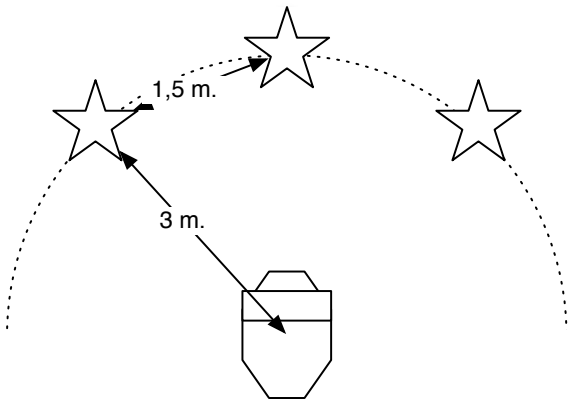


Figure 4.6: Positions of the robot and objects for the collection of the segmentation dataset. The objects were positioned at the starred locations and for each of these locations two orientations of the object was recorded.

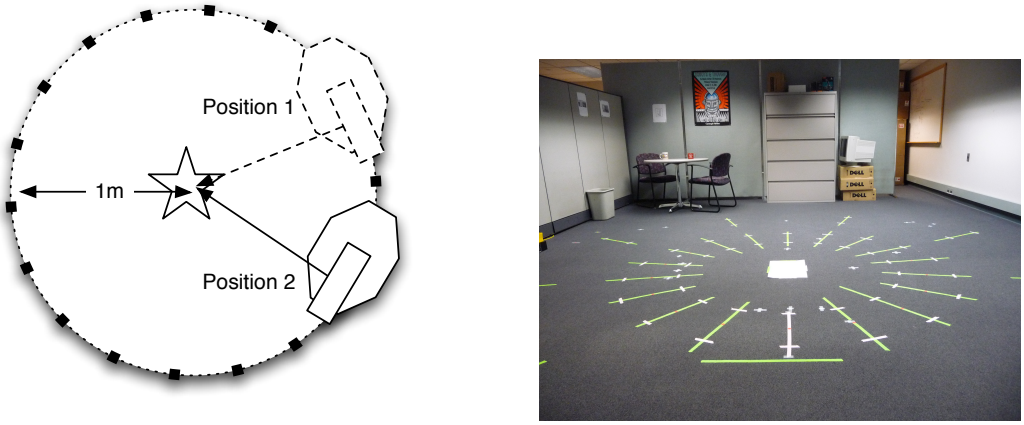
The data was recorded in a resolution of 320x240 pixels, with rectification turned on and using a simple demosaicing algorithm, called "nearest-neighbor". These parameters result in a high framerate, with an average of 9.77 fps. The rectification technique for this dataset has an error; instead of using different camera-parameters for each camera, both cameras use the settings of the left camera. This results in a depth calculation that is of a lesser quality, but it seems that the segmentation techniques are robust against this.

4.2.2 Object Learning Set

This set consists of twenty-one objects, a list of which can be found in chapter B.2. The objects vary greatly in size and texture and all objects can be found in regular office environments. During the recording, the smallest of these were placed on a table to be visible for the robot, when the camera is in

its rest state. This set was made to test the accuracy of creating object models, and the two keypoint filtering techniques (bounding box and active vision).

The robot was first positioned three meters away from the object, with the object in front of the robot. The human experimenter then drove the robot forwards two meters using a remote control, rotating the robot in such a way that the object would be at an angle of 90° left of the robot. The robot automatically aimed its camera using the method described in section 3.2.2, the center of the object was predetermined by the experimenter. The experimenter then proceeded to move the robot to 17 other stops, each time rotating the robot in such a way that the object was at an angle of 90° with respect to the robot's front. At each of these stops the robot's position remained the same for five seconds, with the pan-tilt unit stopping to rotate after approximately two seconds. Two conditions were recorded for each object, one where the distance between the robot and the object was kept at one meter and one where this distance was varied between 50 centimeters to 1.50 meters. For the equidistant recordings, an additional laser-pointing sequence was added of thirty seconds.



(a) The robot was stopped at each black square, the objects were positioned at the starred location.

(b) The set up as seen from the robots starting position

Figure 4.7: The setup for the collection of the single-object dataset and the accuracy measurement set.

A schematic of the robot stop conditions for the equidistant condition can be seen in figure 4.7(a). As can be seen in figure 4.7(b) the 18 stopping positions were marked with green tape. For each position the wheel positions are marked with white tape perpendicular to the green tape and the human experimenter positions the robot on these marks.

Each of the 18 viewpoints differs 20° with the previous one. In comparison, the research of Kootstra et al. (2007) has a similar set-up, but uses 36 different viewpoints meaning that each viewpoint only differed 10° with the previous one. Tests were done with an object database to determine the impact of this decrease in viewpoints on the recognition rates and a difference of 20° in viewing angle was found to still be acceptable (see section 5.2). Having more viewpoints will result in a more stable object model, but it also increases the recording time meaning that less objects could be recorded.

The data was recorded in the highest resolution of the camera, 1024x768, and with a high quality demosaicing algorithm, called HQLinear (Malvar et al., 2004). Rectification was turned off during recording to increase the frame-rate of the recordings. The high resolution and high quality demosaicing algorithm result in a low frame-rate of 1.93 fps on average, but offers good quality images.

4.2.3 Validation Set

This set consists of the same twenty-one objects as in the object learning set, but then in natural settings. It was used to test the robustness of the object models and the accuracy of the recognition

method. Testing on such settings is valuable, since these are conditions under which the robot will be operating. Seven different scenes were recorded with each a number of different takes. Not each scene contains all of the objects, but each object is included in at least two scenes. The objects are scattered in these scenes and can be partially occluded. There are different light conditions for each scene and objects can be obscured by shadows. The distance from the robot to the objects varies. This makes this dataset very realistic, but the conditions make recognition of objects in these sets difficult. From each scene multiple observations were made, each from a different angle. Depending on the complexity of the scene, the robot was moved around to multiple positions in order to get different views of the objects.

Examples of such scenes can be seen in figure 4.8, while all scenes are described in appendix C. As can be seen in figure 4.8, the object occlusion can be quite large, the object 'Book 1' is obscured nearly fully by the object 'Book 2' in scene 6, trial 1 and the 'Red Chair' is nearly cut in half. The influence of shadow can also be seen in this trial. The 'Heater' and 'Disk' objects are completely covered in shadows in this take, for this reason these objects are left out of the annotations (see appendix C). What is also apparent from these images is the influence of the artificial light, nearly all pixels of the 'Pizza Box' in scene 7 trials 1 and 2 are oversaturated.

Scene 1 trial 2

Scene 2 trial 1

Scene 3 trial 1

Scene 6 trial 1

Figure 4.8: Four example scenes used for testing object recognition. In total seven different scenes were recorded, with each one or more trials. This resulted in a total of 17 trials

4.2.4 Accuracy Measurement Set

To measure the accuracy of the odometry in the object learning set, a test set was recorded where the robot drives around a measuring tool, which can be seen in figure 4.9. The robot circles the object in the same manner as described in 4.2.2, using the equidistant one meter condition. The robot is moved by a human experimenter and is halted 18 times for 5 seconds, during each run the robot kept the camera

Figure 4.9: Accuracy measurement tool as used for the creation of the accuracy measurement dataset

(a) Bounding box segmentation (b) GrabCut Segmenation (c) Manual segmentation

Figure 4.10: The two different segmentation techniques ((a) and (b)) with the ground truth (c), these images are cropped from their original 320x240 size. As can be seen in (a) the bounding box has an oversegmentation, while the GrabCut segmentation gives under segmentation (b). This is reflected in the results of experiment 1, in the results section (5.1)

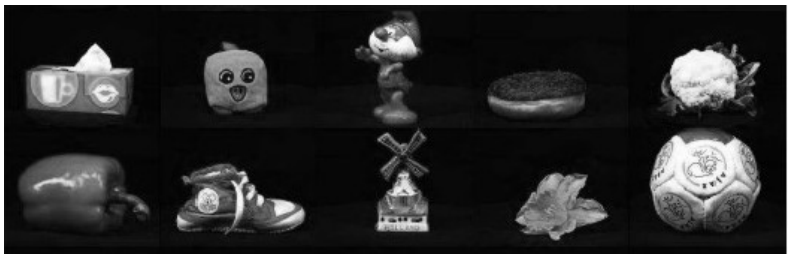
pointed at the object. This was repeated five times.

4.3 Experiments

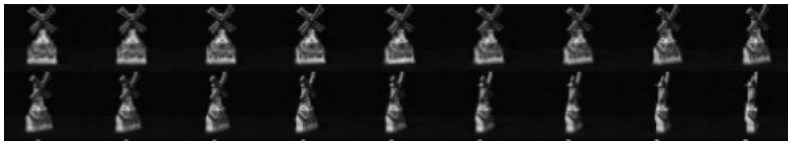
4.3.1 Experiment 1: Accuracy of Object Segmentation

Two different segmentation techniques are described in 3.1, the bounding box and GrabCut. The bounding box is a simple segmentation method based solely on the human designating the object with a laser pointer. The GrabCut segmentation method is a refinement on this simple technique and it uses color and edge information. The successful creation of an object model depends on this initial segmentation. If too much background information is included, or if too much of the object is discarded the object model will be less accurate.

Both segmentation techniques were tested with the use of the object segmentation dataset (see section 4.2.1). To test the accuracy of these methods object masks were created manually, for each object, position, pose and background. An example of such a mask can be seen in figure 4.10(c). To measure the performance of the algorithms, the True Positive Rate (also called precision) (TPR) and False Positive Rate (also called recall) (FPR) were calculated for each algorithm, using the manually created mask as ground truth. A true positive is defined as a pixel that is marked foreground in both the ground truth and the test mask, a false positive is a pixel marked as foreground in the test mask but marked background in the ground truth. Similarly, true negatives are pixels marked background in both masks, while false negatives are pixels marked background in the test mask but marked foreground in the ground truth. The TPR and FPR are then calculated as follows:



(a) Example objects



(b) Example in viewpoint differences (0° to 90°, with steps of 5°)

Figure 4.11: Examples of the "Amsterdam Library of Image Objects". Images from Geusebroek et al. (2005)

$$TPR = \frac{|TP|}{|FN| + |TP|} \tag{4.1}$$

$$FPR = \frac{|FP|}{|TN| + |FP|} \tag{4.2}$$

With $|TP|$ being the number of pixels labeled True Positive (TP), $|FN|$ the number of pixels labeled False Negative (FN), $|FP|$ the number of pixels labeled False Positive (FP) and $|TN|$ the number of pixels labeled True Negative (TN).

4.3.2 Experiment 2: Robustness of SURF against Changes in Viewpoint

For the object learning set 18 viewpoints were used, each viewpoint differing from the previous one by up to 20°. In comparison, in the research of research of Kootstra et al. (2007) twice as many viewpoints were used, the change in viewpoint being only 10°. The active vision approach discards keypoints based on whether they appear in the next and previous viewpoint, meaning that the change in viewpoint cannot be too large. To measure what the limitations of the SURF matching capabilities are under different viewing angles, an experiment was set up where the viewpoint was methodologically changed with a difference of 5°.

To test the influence of a change in viewpoint the "Amsterdam Library of Image Objects" Geusebroek et al. (2005) (ALOI) wast used. This dataset contains thousand small objects, recorded with viewpoint changes of 5°, under different light directions and illumination colors. The set contains a diversity of objects, such as toys, house hold material and fruits and vegetables (see also figure 4.11(a)). The different light and illumination color conditions were ignored for this experiment, the resolution of the images used was 384x288. All objects were thus illuminated in the same way, by five lights and under the same color conditions. The object was rotated on a rotation table with steps of 5° as can be seen in figure 4.11(a).

From this dataset only those objects were used from which more than 40 SURF-keypoints could be extracted in the first viewpoint (0°) of the object. Objects with a lower number of keypoints are too texturally sparse, meaning that an object model created with SURF would be unsuitable for storing such an object. After filtering these objects, 523 objects were left.

The influence of a difference in viewpoint to an object on the matching capabilities of SURF, was measured by calculating the percentage of keypoints that could still be matched between different viewpoint changes n, where $n \in \{5°, 10°, 15°, 20°, 25°, 30°\}$. A match was defined as a keypoint which fell

within a euclidean distance of 0.7 of a keypoint in a different viewpoint, based on the recommendation of Bay et al. (2006). For each of the images belonging to a change in viewpoint a set of keypoints $(S_5, S_{10}, S_{15}, S_{20}, S_{25}, S_{30})$ was extracted and the percentage of keypoints that could still be matched from viewpoint 0° was calculated, as follows:

$$D(S,k) = \begin{cases} 1, & \text{if } \min_{p \in S}(||k-p||) < 0.7 \\ 0, & \text{if } \min_{p \in S}(||k-p||) > 0.7 \end{cases} \tag{4.3}$$

$$P(n) = \frac{\sum_{k_i \in S_0} D(S_n, k_i)}{|S_0|} \tag{4.4}$$

As described in section 3.2.2 keypoints are matched in view $n-1$ and $n+1$, and not only in $n+1$ as is the case in the experiment described above. To see the influence of matching in two different views another experiment was run, where for each keypoint in S_0 the distance was calculated to keypoints in two sets of keypoints. Resulting in the following score, with $n \in \{5°, 10°, 15°, 20°, 25°, 30°\}$ and S_n being the set of keypoints belonging to the image at viewpoint n:

$$R(n) = \frac{\sum_{k_i \in S_0} D(S_n \cap S_{(360-n)}, k_i)}{|S_0|} \tag{4.5}$$

4.3.3 Experiment 3: Accuracy of Odometry and Stereo Vision

During object learning the robot will actively explore the object. It uses its odometry to keep the camera pointed towards the object centre and to transform the pointcloud from the stereo vision camera to an absolute world model with the origin on the starting point of the robot. During the learning phase keypoints are filtered based on their position in the absolute world model. There is a combined error of the odometry, depth estimation from stereo vision and the rotation transformation of the points due to pan and tilt (the pan tilt unit has a limited angular resolution, but the orientation it reports is not limited to this resolution) and in this experiment this error is measured.

For each stopping position i in the accuracy measurement set, the position in 3D space of the red dot $\vec{d_i}$ on the accuracy measurement tool is determined using the same methods to determine positions in space that are used for the object learning condition. The measurement $\vec{d_i}$ is stored in the set of measurements D. The combined error of the odometry, stereo vision and the rotation transformation of the points due to pan and tilt can now be measured by calculating the Euclidean distance of the estimated position of the dot with respect to the position of the dot in the first measurement $\vec{d_1}$. As the dot itself has a diameter of 1.60 centimeter instead of being an absolute point in space, the measurement is not completely precise, but compared to the error of the odometry this imprecision is negligible. The error $E(D)$ is calculated as follows, with D being the set of calculated positions of the red dot on the measurement tool:

$$E(D) = \frac{\sum_{(\vec{d_i} \in D) \wedge (\vec{d_i} \neq vecd_1)} ||\vec{d_1} - \vec{d_i}||}{|D|} \tag{4.6}$$

4.3.4 Experiment 4: Accuracy of Object Recognition

To test how well the recognition method works as well as how well the creation of the object models goes, the accuracy of these methods were tested by evaluating several object models created from the object learning dataset. The equidistant recordings were used to create object models for each of the 21 objects. These models were tested under two conditions, the first condition uses the views in the "random distance" object learning dataset as test set, the second using the "scene" recordings as test set. These datasets are described in section 4.2.2 and section 4.2.3 respectively.

Condition one: Random distance

Two types of object models were created using the equidistant recordings from the single-object set. The first was the normal object model as described in 3.2.3, designated as "multiple views" models (the set of all these models is $\mathcal{A}$). To compare the performance of this model, a one shot learning model was created for each object as well (the set of all single view models is $\mathcal{I}$). This object model consists of only one of the views of the "multiple views" model. The recognition method was the same for both types of object model and is described in section 3.3. Both object models had their keypoints filtered using the active vision method and the bounding box method. The recognizer $R(S, \omega)$ returns the highest scoring object ID, based on a set of object models S and a set of keypoints ω. Both models were tested using the random distance recordings from the object learning set. The performance was calculated per object by running the recognizer on all of the views of this object in the random distance dataset. The objects were not segmented in the views, leaving the object in a complex environment with a complex background. The performance $p(S, ID, \omega)$ on a single view ω of object ID, using a set of object models S is defined as follows:

$$p(S, ID, \omega) = \begin{cases} 1, & \text{if } R(S, \omega) = ID \\ 0, & \text{if } R(S, \omega) \neq ID \end{cases} \tag{4.7}$$

For the "multiple views" model the score was calculated as the percentage of correctly classified views of the object. The performance $P_{\mathcal{A}}$ is determined per object ID, by determining the percentage of views $\omega \in \Omega_{ID}$ of the random distance dataset that were correctly classified as the object ID.

$$P_{\mathcal{A}}(ID) = \frac{\sum_{\omega \in \Omega_{ID}} s(\mathcal{A}, ID, \omega)}{|\Omega_{ID}|} \tag{4.8}$$

The performance $P_{\mathcal{I}}$ of the comparison method was calculated slightly differently by taking the average of the performance score on each of the views. For each view, the recognizer R was run using all views from the "multiple views" model for that object ID.

$$P_{\mathcal{I}}(ID) = \frac{\sum_{\omega \in \Omega_{ID}} \frac{\sum_{k \in \mathcal{A}_{ID}} s(\mathcal{I}, ID, \omega)}{|\mathcal{A}_{ID}|}}{|\Omega_{ID}|} \tag{4.9}$$

Condition two: Real world scenes

To test the performance of this method in the real-world, the recognizer was also run on the validation set. For this condition, the recognizer was only used with the multiple views object models, but the settings in creating this model and running the recognizer were varied.

The validation set consists of scenes, which in turn consists of one or multiple takes. Each take t is defined as the image I_t and the set $\mathcal{L}_t$ of objects in this image. For each I_t the recognizer returns a list of objects ranked by activation, with $\mathcal{R}_t$ being the first n_t objects in this list, with $n_t = |\mathcal{L}_t|$.

The performance of the recognizer on a take t then becomes:

$$P_T(t) = \frac{|\mathcal{L}_t \cap \mathcal{R}_t|}{n} \tag{4.10}$$

The performance of the recognizer on the entire set of scenes S is then:

$$P_S(S) = \frac{\sum_{t \in S} |\mathcal{L}_t \cap \mathcal{R}_t|}{\sum_{t \in T} |\mathcal{L}_t|} \tag{4.11}$$

To test the influence of the two filtering methods during the learning stage, using a bounding box and using active vision, the recognizer was also run using object models where these methods were turned off during the learning stage. The filtering technique by Lowe (2004) during recognition was tested in a similar way, by running the recognizer with it either turned on or off.

Chapter 5

Results

5.1 Results of Experiment 1: Accuracy of Object Segmentation

The results of the experiments as described in section 4.3.1 were averaged over all objects, poses and positions and can be found for both backgrounds and segmentation methods in table 5.1 and figure 5.1[1]. As can be seen from table 5.1, the bounding box method has a high level of correctly classified object pixels (TPR). This simple segmentation technique works remarkably well, but this is also due to the choice of objects. Most of the objects can be defined as a box like shape (with the exception of object 2, the 'Table', see figure 4.3(2)). For both backgrounds the TPR of the GrabCut segmentation technique is significantly worse than the Bounding Box technique (using a Student T-Test $p << 0.01$ for background 1, $p << 0.01$ for background 2, $\alpha = 0.05$).

The simple segmentation technique works less well for classifying background pixels, the FPR is much higher then that of the GrabCut method. Note that the FPR is considerably lower than the TPR due to the way these are calculated, the term below the division for FPR is much higher than that of the TPR. This is because the number of true negatives is very large as the object is only a small part of the image. For the first background the error is nearly three times as high for the bounding box method as for GrabCut. In the second background there is an even higher discrepancy between the false positive rate of the bounding box method and GrabCut, it is nearly ten times as high. These differences were also tested with a Student T-Test and they are again highly significant for both backgrounds ($p << 0.01$ for background one, $p << 0.01$ for background two, $\alpha = 0.05$).

There is also a very apparent difference between the different backgrounds, as both object segmentation techniques work better in the first background condition. This is very likely more due to the difference in lighting conditions than the difference in background, as it was to be expected that the GrabCut segmentation would work better with a clean background (which is the second background). In the dataset it is easy to see that a lot of the object and background pixels in the images of the second background are oversaturated.

Background	Bounding Box TPR	Bounding Box FPR	GrabCut TPR	GrabCut FPR
One	$\mu = 0.943$	$\mu = 0.033$	$\mu = 0.723$	$\mu = 0.012$
	$\sigma = 0.138$	$\sigma = 0.028$	$\sigma = 0.250$	$\sigma = 0.014$
Two	$\mu = 0.854$	$\mu = 0.123$	$\mu = 0.590$	$\mu = 0.013$
	$\sigma = 0.294$	$\sigma = 0.146$	$\sigma = 0.320$	$\sigma = 0.015$

Table 5.1: The average true and false positive rates for the two segmentation methods, averaged over all objects, positions and orientations

[1]For the original data, please see: `http://www.ai.rug.nl/~zwinder`

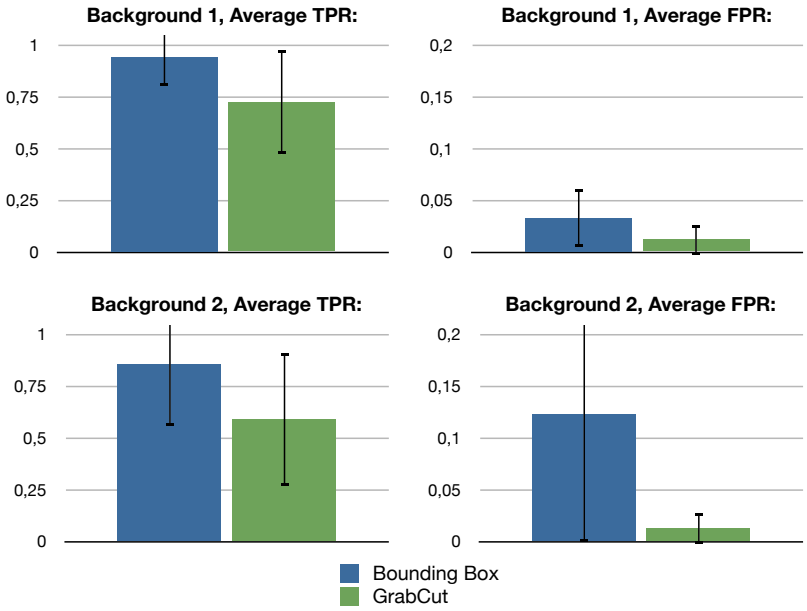


Figure 5.1: The average true and false positive rates for the two segmentation methods, averaged over all objects, positions and orientations. Note that the axis are not the same for the left and right figures

5.2 Results of Experiment 2: Robustness of SURF against Changes in Viewpoint

As can be seen in figure 5.2 the percentage of keypoints that can be matched from the first viewpoint (0°) decreases as the viewing angle increases, which can be expected as some keypoints can be self-occluded by the object when changing the viewpoint or keypoints are not invariant or robust enough against a higher change in viewpoint. If the change in viewpoint becomes larger, the active vision filtering approach (described in section 3.2.2) will discard more keypoints and if the change in viewpoint becomes too large the method will discard too many keypoints, meaning that the resulting object model will not be correct. The active vision filtering method is based on Kootstra et al. (2007), and in that research the change in viewpoints was limited to 10°. In the dataset used for object learning in this research (see section 4.2.2), the change in viewpoint is 20°. As can be seen in figure 5.2, such an increase in change of viewpoint (20° versus 10°) results in a large decrease in matching keypoints. When using only one reference viewpoint, as in Kootstra et al. (2007), this means that only 31% of the keypoints would remain after filtering with a viewpoint change of 20°. This is a large difference compared to the percentage of matching keypoints with a viewpoint change of only 10°, in this case an average percentage of 49% keypoints can match. This difference is highly significant (T-Test, $\alpha = 0.05, p << 0.01$).

However, as can be seen in the graph, using two reference viewpoints will partially solve the problem of discarding so many keypoints. By using two reference viewpoints nearly 45% of the keypoints can still match at 20°, which is only a slight but significant, decrease when compared with the percentage of matching keypoints in the single reference viewpoint situation at 10° (T-Test, $\alpha = 0.05, p << 0.01$). In comparison, in Kootstra et al. (2007) the percentage of matching keypoints between two views with a difference of 10° was 36% and filtering this amount of keypoints still resulted in a better recognition accuracy. The difference in percentage of matching keypoints can be found in different matching procedures (PCA-SIFT versus SURF) and recording conditions (the object in ALOI is segmented by using a very simple environment, while the objects in the dataset of Kootstra et al. (2007) are situated in a more cluttered environment).

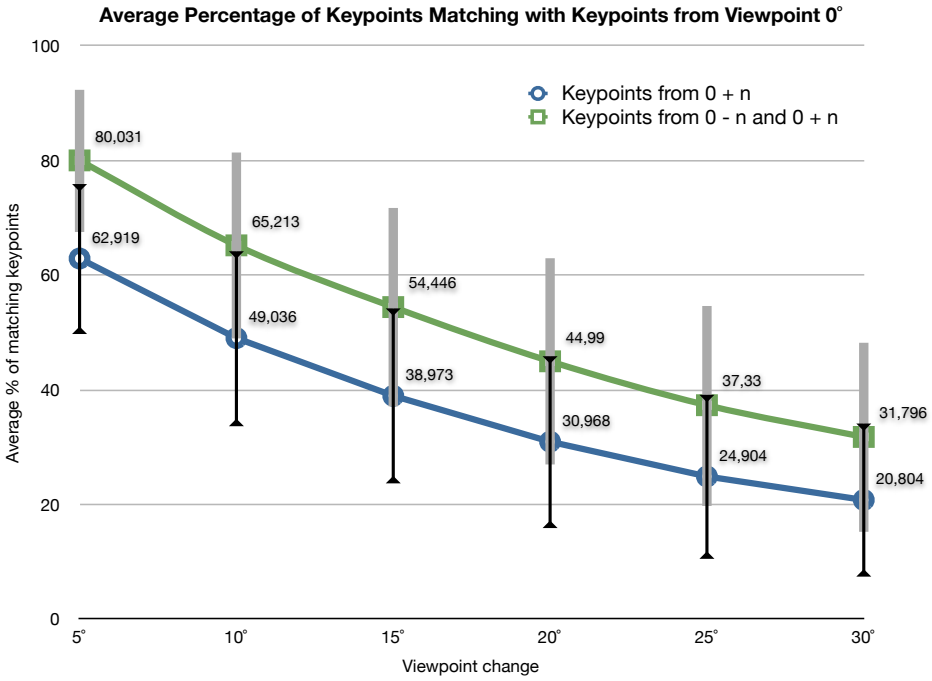


Figure 5.2: Results of Experiment 2. For each change in viewpoint $n \in \{5°, 10°, 15°, 20°, 25°, 30°\}$, two sets of keypoints were extracted $V(n)$ and $V(n) \cap V(-n)$. The percentage of keypoints from $V(0)$ that could be matched with these sets can be seen plotted in the above figure for different n. This was done for 523 objects in the ALOI dataset. Here the average matching percentage is shown, with the error bars showing the standard deviation. By using two reference viewpoints the percentage of matching keypoints can be greatly increased.

5.3 Results of Experiment 3: Accuracy of Odometry and Stereo Vision

As the bounding box filtering method works based on the position of keypoints in the absolute world model (as described in section 3.2.2), the error of calculating the keypoints' position cannot be too large, as keypoints belonging to the object might get incorrectly discarded. The error was measured as described in section 4.3.3, over all five recordings as described in section 4.2.4. The average error over all stopping positions, in all recordings is 21.47 centimeters. ($\sigma = 9.89$). In many conditions the error margin in the bounding box will be large enough to not discard object keypoints incorrectly, however an improvement on this error would be welcome. Especially for small objects such a large error may cause problems, as the margin around the bounding box is calculated as a percentage of the object size.

5.4 Results of Experiment 4: Accuracy of Object Recognition

As described in section 4.3.4 the accuracy of the object models and the recognition method was tested with two experiments. Both experiments use the equidistant object recordings (as described in section 4.2.2 as learning set, but they use different datasets as test sets. The first experiment uses the random object recordings (also described in section 4.2.2) as test set, while the second uses the scene-based recordings (section 4.2.3).

5.4.1 Condition 1: Random distance

As described in section 4.3.4 the first experiment used two types of object models for the recognizer. One object model used only one view and one object model uses multiple views. Both models were used as object models for the recognizer to recognize an object from its different viewpoints in the random distance condition of the object learning dataset. The score is then determined as the percentage of views that are correctly recognised as being the object. The single object model was run multiple times on each viewpoint, each time using a different view from the multiple views model. The multiple views model was only run once on each viewpoint of the object, which is why this object model has no error bars in the graphs.

The percentage of correctly classified views per object can be seen in figure 5.3. From this figure, it is apparent that the performance of the multiple views model is much better than that of the single view model. A one-sided Student T-Test shows that these differences are significant ($p << 0.01$, with $\alpha = 0.05$). This shows that using a multiple views model has a huge advantage over using a single shot learning model.

Although the multiple views models result in a higher performance score, these results can be improved upon, as can be seen in figure 5.3. Only eight objects out of 21 objects are recognised consistently in all of the views, being recognised in over 60% of the cases. This may not look like much, but it is very promising, considering that these objects are diverse in size, structure and texture and that the recording of both the test and the training set was done in a real-world setting. However, several other objects are recognised poorly. Both the 'Big Chair' and the 'Cap' objects can not be recognised by the multiple views model in any of the views. The 'Cap' object is texturally very sparse and due to it being small it generates less keypoints. As the bounding box has a safety margin of 20%, some keypoints that belong to the table were also added to the 'Cap'-object model. Furthermore, due to its bright white color, the CCD-sensors were oversaturated for some views of the 'Cap'. Other objects that are recognized poorly, such as the 'Cup', the 'Chocolate Box 2' and the 'Disk' are small in size as well. The 'Chair' object is the opposite, it is richly textured, large and not reflective. Why the object model fails for this object cannot be said for certain, the texture might be too fine grained to create descriptive keypoints for instance (the object-model does not contain less keypoints than that of other objects). Other objects that are recognised poorly that are also big are the 'Red Chair' and the 'Table' and both have little texture.

The performance of the recognizer has room for improvement and to see how close the recognizer is to recognizing the correct objects, the performance is measured by judging the top n matches as well versus only looking at the top result ($n = 1$). By taking $n = 2$ and $n = 5$ the performance greatly

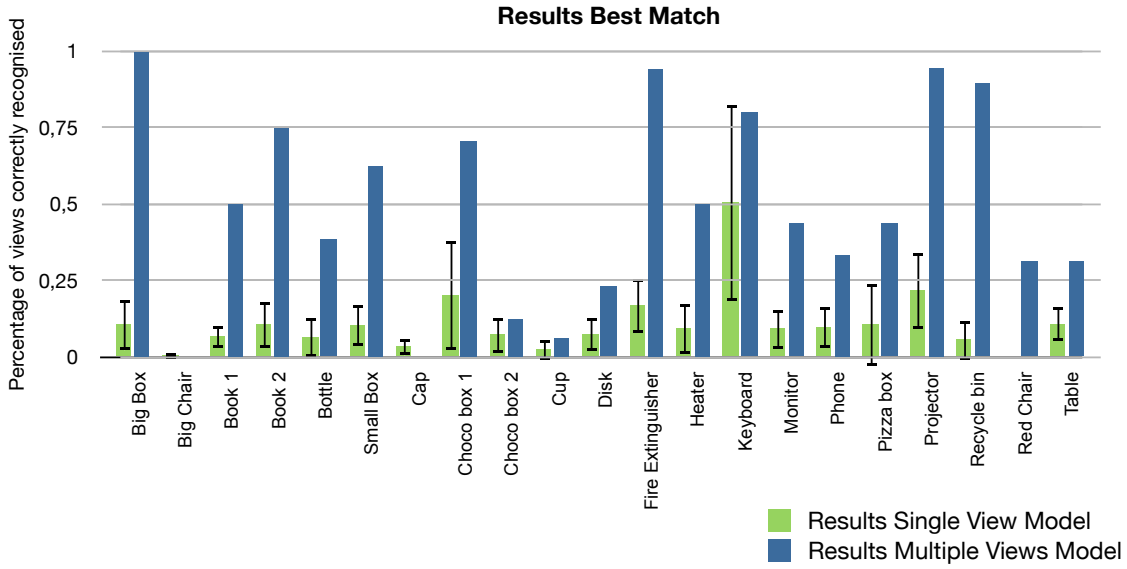


Figure 5.3: Two recordings were made of each object, one at a 1 meter distance to the object and one at a random distance to the object, where the robot stopped at several positions. From the first condition (1 meter equidistant), two object models were made, a single-view model and a multiple-views model. To test the performance of each type of model, the recognizer was run on each viewpoint in the random distance condition. This figure shows the percentage of viewpoints that were recognised as the correct object. The single-view model was run on each viewpoint multiple times, using a model based on a different viewpoint from the 1 meter condition. Where the match for the multiple views model per viewpoint is either 0 or 1, the single view model can thus have a score $\{0, \cdots, 1\}$. The error bars show the standard deviation of these and are therefor not present for the multiple views model. As can be seen in the figure the multiple views model performs better for all objects.

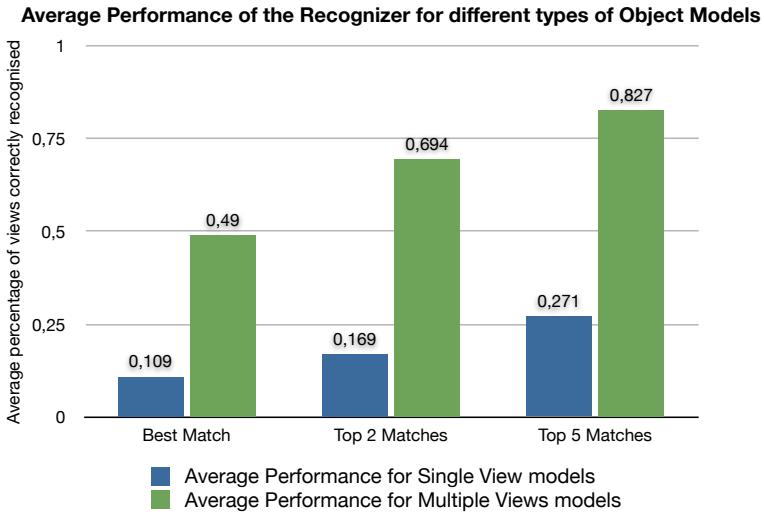


Figure 5.4: The average percentage of viewpoints recognised as the correct object of the random distance condition in the object learning set. For the "Best Match" condition only the first matching object returned by the recognizer was considered. For the "Top 2 Matches" the two highest scoring objects were considered and a correct recognition was said to have taken place if the object was in this list, similarly for the "Top 5 Matches". As can be seen from this figure the multiple-views model scores better than the single-view model. As can also be seen in the figure, the correct object is often in the top of the recognizers results. This suggests that there is room for improvement in the approach.

increases, as seen in figure 5.4. These results show that the performance of the recognizer can be further improved if it is possible to discard some of the false positives. In comparison, if the recognizer would simple be guessing for the $n = 5$ condition, the performance would be circa 0.25. With this in mind the single view model scores very bad and the multiple views model scores well above chance level.

Taking a more in depth look at the individual results per object in figure 5.5, it is clear that by even looking at only the top 2 results there is an improvement. For several objects ('Book 1', 'Choco Box 2', 'Cup', 'Disk', 'Phone', 'Red Chair' and 'Table') the improvement in recognition accuracy is more than 50%, meaning that they are correctly recognised in more views. Where as the 'Cap' object is never present in the best match condition, it is correctly recognised in 55% of the views when looking at the top 2 matches. To see the influence of these false positives a frequency histogram was made for the false positives. For each incorrect match at $n = 1$ the false positive was noted, resulting in the diagram in figure 5.6. As is clear from this diagram, the 'Big Box' object is most often incorrectly given as response. This means that the 'Big Box' object may contain very general keypoints that match for many objects. In 52 of the 112 cases that the recognizer incorrectly returns 'Big Box' , the correct object ID is the second scoring object ID. This shows again that an improvement can be made relatively easily by discarding false positives.

5.4.2 Condition 2: Real world scenes

As described in section 4.2.3, this experiment tests the performance of the recognizer on different real-world scenes. The scenes differ in lighting condition and viewing angle and the distance to the objects varies, objects can be partially occluded and can be covered by a shadow. The performance of the recognizer is measured as the percentage of objects that are correctly recognised over all the scenes. Besides this the settings of the recogniser were varied over the different filtering methods.

From figure 5.8 (and the data in table 5.2), can be seen that using both filtering techniques during learning results in the best performance. The filtering method of Lowe (2004) improves the recognition

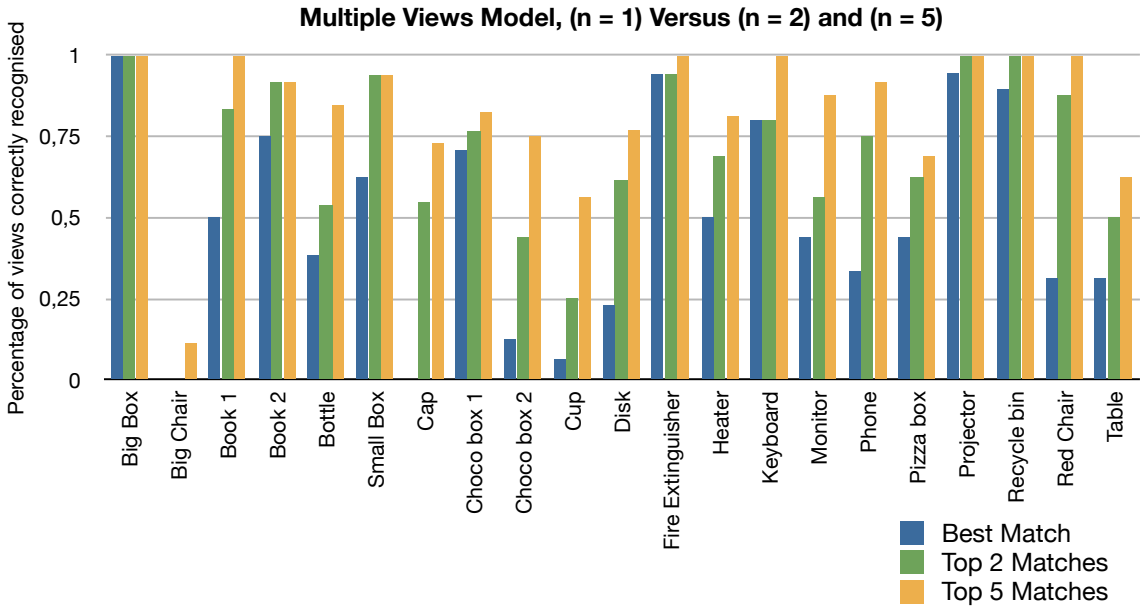


Figure 5.5: Multiple views model, best match, top 2 matches and top 5 matches. In this figure the recognition percentage per object can be seen. As can be seen, here it is obvious that the recognition rate increases significantly for some objects when considering not only the best, but also the top-2 or top-5 matching results. For instance, the "Cap" object is not recognised correctly once in the "best match" condition, but is found in the top-2 and top-5.

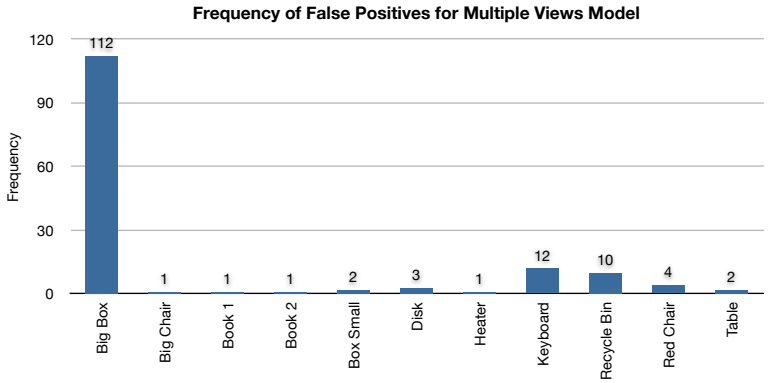


Figure 5.6: Frequency histogram of the false positives for the multiple views model, in condition $n = 1$ ("Best Match"). As can be seen, most of the false positives are the "Big Box" object. This suggests that an improvement of the recognition rate is possible by specifically checking for specific object-features belonging to false positives

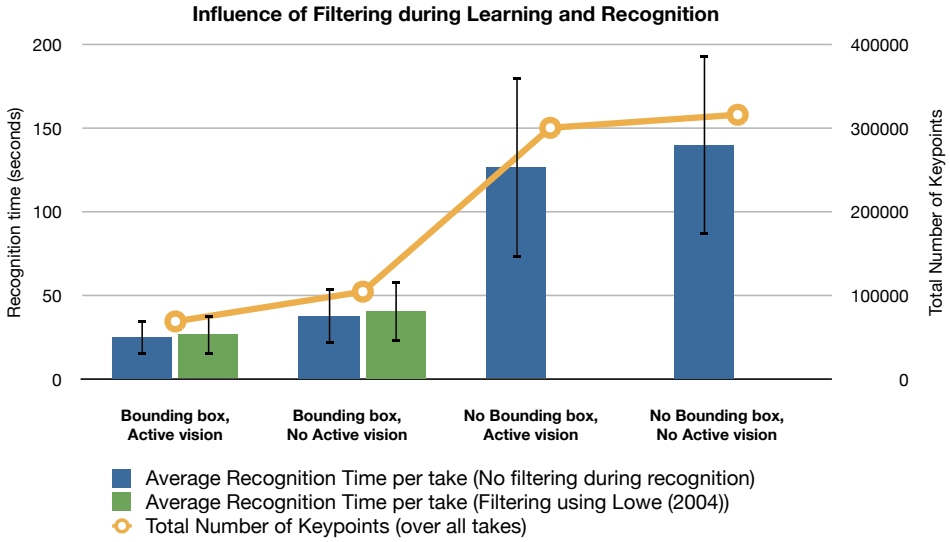


Figure 5.7: Recognition times averaged over all takes, for several configurations of the system. The error bars indicate the standard deviation of the average recognition times. The "Total Number of Keypoints" is the total amount of keypoints in all object models, after filtering. Under the two right most conditions, the filtering-method using Lowe (2004) could not be run within the time frame of the project, due to the high amount of keypoints. The "Bounding Box" and the "Active Vision" filtering techniques are explained in section 3.2.2, while the filtering during recognition based on Lowe (2004) is explained in section 3.3. From this figure it can be seen that the filtering methods have a high impact on the number of keypoints that are stored and that this in turn influences the recognition times. As can be seen, the amount of keypoints that is filtered using the "Bounding Box" method is significantly larger as the "Active Vision" filtering technique.

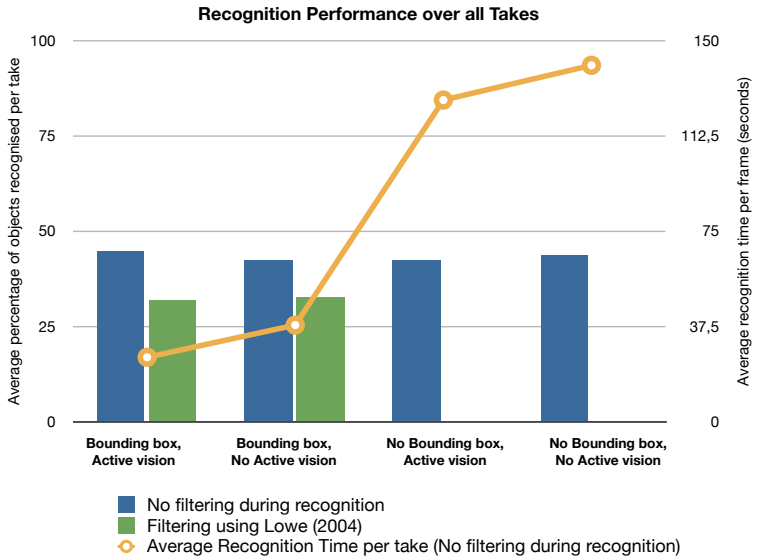


Figure 5.8: Recognition rates for all objects in the test-scenes, for several configurations of the system. The recognition rate is determined as the percentage of objects that is correctly recognised in each take. Each scene contains a complex configuration of objects, with partial occlusions, shadows and differing light conditions. The recognition rate is therefor not high. What can be seen from this figure is that the recognition rate remains constant even after removing many keypoints by filtering. This shows that the filtering techniques do not remove any significant keypoints. The drop in recognition time is very large.

rate as well. However, the difference between using filtering during recognition and learning versus no filtering, the largest difference in recognition performance, is not significant (using a one-sided T-Test, $\alpha = 0.05$, $p = 0.1663$). What is also apparent is that the recognition rates are not high, slightly less than 50% of the objects can be recognised. However, the recording conditions are very much a real-world setting and the light conditions differ a lot from the training-data. In certain conditions the objects' view is obstructed in a large way by other objects or are only partially visible by the camera. Another major influence are the shadows.

As can be seen in figure 5.7, both filtering methods during the learning stage greatly influence the number of keypoints that are stored. This number of keypoints in turn influences the recognition times. Using both filtering techniques results in a decrease of keypoints by 4.6 compared to using no filtering. This in turn results in a recognition time per take that is 5.5 times less. Even though recognition is not yet in real-time (taking 26.718 seconds on average per take, when using filtering during learning and recognition), the filtering techniques do make a large difference in recognition time. The filtering technique during recognition does not add much additional processing time for the first two conditions, but results in a massive slowdown for the second two conditions. After multiple days, the filtering was still not complete for one take in both conditions. These results are thus not present in the tables. What can clearly be seen is that reducing the number of keypoints for an object model in the learning stage by the two filtering techniques does not impact the recognition rate negatively. The amount of data that needs to be processed is several factors less, but the recognition rate is higher, which is a very impressive result.

Filtering during learning:		Filtering during recognition:	
Bounding box	Active vision	None	Using Lowe (2004)
Yes	Yes	$44,6\%$	$47,8\%$
Yes	No	$42,4\%$	$47,8\%$
No	Yes	$42,4\%$	—
No	No	$43,5\%$	—

Table 5.2: Recognition rates for all objects in the test-scenes, for several configurations of the system

Chapter 6

Discussion

6.1 Summary of the Results

By using a laser pointer objects can be referred to and segmented easily by a human. A robot companion can use this designation and segmentation as a starting point for the creation of an object model. The accuracy of the laser-recognition method described in this work is such that segmentation can be done very successfully.

By using this segmentation an object model can be created which contains far less data. The amount of useful information in the object model remains the same. In addition to this filtering technique active vision can be used by the robot. Due to the human reference the robot knows the position and size of the object and can successfully keep it's camera orientated towards the object centroid. This filtering technique further decreases the amount of data in the object model.

Both filtering techniques work successfully in the reduction of data, while keeping the recognition rate high. The learning and recognition methods described in this work were tested in a real-world situation, meaning that the performance rates are representative of what can be expected in future applications.

6.2 Solving Deictic Reference and Initial Object Segmentation

It is shown that by using a laser pointer it is possible to establish a deictic reference with high precision in a real-world environment. It is shown that the device can be used as a tool in object teaching, with which a human can present the robot with an initial object segmentation in image- and three-dimensional-space with great accuracy as shown in section 5.1. This was tested with a test-set of seven objects, and the true positive rate of the segmentation is 85.4% to 94.3% in different real-world office environments. This is an extension on the work of Kemp et al. (2008) where the user-robot interaction with a laser pointer was limited to object referral as done in Kemp et al. (2008). As shown in that research conducted by Kemp et al. (2008) the use of a laser pointer is not dependent on a specific person.

The use of a laser pointer is helpful in human-robot interaction and it has several advantages over other methods. The laser pointer provides the human with direct visual-feedback object referral and segmentation, something which is lacking from natural human gestures (Horn and Ward, 2004). In addition to this, it does not suffer from the considerable error that is present when trying to resolve natural pointing (Ziegler et al., 2006). Because of this, there is no need for the human to be in close proximity to the object when using a laser pointer, as is the case in many other robot systems featuring human-robot interaction for object learning (Ghidary et al., 2002, Lomker and Sagerer, 2002, Moller et al., 2005, Toptsis et al., 2004). The laser pointer is thus preferred over such other methods. The laser pointer does not share the limitations of other assistant pointing devices (Patel and Abowd, 2003, Wilson and Pham, 2003, Wilson et al., 2003), because there is no limitation on a specific room or a limitation on pre-tagged objects.

The current laser pointer detection is based off of similar systems used to detect a laser pointer during presentations (Kirstein and Mueller, 1998, Olsen Jr and Nielsen, 2001) or to designate objects to

a mobile robot (Kazi et al., 1995, Kemp et al., 2008). As in most of these methods, a combination of motion detection and color thresholding is used. The currently deployed method extends these methods by applying clustering to detected laser points. The system uses stereo vision to apply a position in three-dimensional space to each detected laser point and detected laser points can be discarded if they are not part of an object cluster. This extension provides a means to discard false detections not present in the other methods.

The laser-based human segmentation can optionally be extended by using the computer-vision technique GrabCut (Rother et al., 2004). As the preliminary segmentation is a bounding box, many of the pixels not belonging to the object are falsely found in the foreground selection, GrabCut is designed to enhance such a segmentation using both color and contour information (Rother et al., 2004). The percentage of false positives is indeed decreased significantly by using this technique versus using only the bounding box, however, the percentage of true positives is also decreased significantly. This means that the GrabCut segmentation technique does not offer an advantage over using only the bounding box approach.

Improvements

One promising technique that can be used to improve the segmentation of the bounding box is SnakeCut, a method showing better segmentation-results than other techniques including GrabCut (Prakash et al., 2006). This technique combines Snakes (Kass et al., 1988) and GrabCut, increasing the accuracy of segmentation in many cases (Prakash et al., 2006). Furthermore, depth information can be used to enhance this segmentation. The predecessor to GrabCut, called Graph Cut is already being used for this (Kolmogorov and Zabih, 2002), meaning that such an enhanced can be made. By using Graph Cut not only the object segmentation is enhanced, stereo depth information in the overall image will increase (Kolmogorov and Zabih, 2002). Currently, the stereo vision algorithm is a simple pixel based approach (as detailed in appendix A) which uses only information from the current frame and improvement can be made on this front.

The segmentation of the object in three-dimensional space is done by taking the width and height of the bounding box segmentation, the depth of the object is estimated to be the same as the width. This approach works fine for the objects tested, but might pose problems for larger, not uniformly shaped objects. An example of these is a couch. The object model should in this case grow while the robot moves around the object, instead of being confined to the preliminary bounding box. This is currently not implemented.

6.3 Object Learning and Active Vision

By making use of active vision, the object models are more robust against viewpoint changes as can be seen in the result of experiment 4 in Section 5.4. There is a highly significant improvement over using a multiple views model versus a single view model, as was to be expected from literature (Dutta Roy et al., 2004, Kootstra et al., 2007).

The active vision approach used in Kootstra et al. (2007) is successfully extended in two ways, by using two reference views ($n-1$ and $n+1$) instead of one and by removing the need for a circular path. By using two reference frames the method is robust against a larger difference in viewpoints, 20° instead of 10° as shown in the results of experiment 2 in Section 5.2. With the approach of Kootstra et al. (2007), the robot needs to be placed in front of the object, with its camera pointed at the object. The robot then proceeds to make a circle, using field of flow to discard keypoints belonging to the background. The camera does not actively track the object, the approach instead relies on a good alignment of the robot and the camera in the starting position. The method works best when no other objects are close to target object and can not account for deviations of the path. The current approach using the initial human selection does not have these limitations.

Improvements

Currently the robot is moved by the experimenter in all experiments and as such an obvious next step is to have the robot control its own path. Because of the changes made to the original method of Kootstra et al. (2007) this path does not need to be a circle. The current system controls the camera angle by itself, tracking the object centroid automatically and the system automatically decides whether a current location is suitable for taking a stable image. Because the size and position of the object are known before starting the learning process, it should be possible to add path planning relatively easy.

The use of odometry in the position estimation of the robot introduces a large error, as can be seen in the results of experiment 3, sections 5.3. The average error is 21.47 centimeters, in conditions similar to the learning object phases. This error can be reduced by using a more detailed world model, based for instance on (visual) Simultaneous Localization And Mapping (SLAM) (Bailey and Durrant-Whyte, 2006, Smith and Cheeseman, 1986). By using SLAM a map is built while driving around and by using statistical methods different data (such as odometry and depth from stereo-vision) can be combined to reduce the error of the position estimation.

6.3.1 Robustness of SURF Against Viewpoint Changes

SURF keypoints are shown to be robust against various viewpoint changes, but the percentage of matches decreases with an increase in viewpoint change. Moreels et al. (2007) researched the decrease in matching keypoints for a change in viewpoint as well. Several different descriptors (such as Scale-Invariant Feature Transform (SIFT) and Principal Component Analysis (PCA)-SIFT, but not SURF) were tested and the matching percentages seem equal to the results here. According to Moreels et al. (2007) the matching percentages differ per object, for irregular structures such as toy cars the change in viewpoint can only be minimal before the percentage of matches becomes small. For such objects a change of 20° results in less then 20% recognition rate for all methods tested. The stability of keypoints extracted from flat objects such as boxes is significantly higher, ranging up to 40% recognition rate for different methods. In comparison with the methods researched in Moreels et al. (2007), SURF seems to perform above average. A direct comparison cannot be made, as the datasets are not the same.

Kootstra et al. (2007) also use a mobile robot and active vision to learn object representations, using the SIFT feature-selection and description-method. Instead of 20° difference between viewpoints there is only a 10° difference, resulting in a decrease of keypoints of 36%. Using only one matching viewpoint with SURF, the number of matching keypoints is 49%, suggesting that SURF is more robust against such viewpoint changes. However, the recording conditions between Kootstra et al. (2007) and the ALOI dataset are not completely comparable. Furthermore, in Kootstra et al. (2007) keypoints are also discarded based on flow of field, to discard background pixels while such a method is not used in the current method. However, as the objects in the ALOI-dataset do not have a background this does not seem to pose a problem.

6.4 Object Recognition

As shown by both experiments in section 5.4, by using both the learning and recognition methods described in chapter 3, objects can be recognised in real world situations with a reasonable accuracy. By using the two filtering techniques during learning the recognition time can be optimized greatly, without affecting the recognition accuracy. It is shown that the methods are not limited to one type of object, but instead work for a large range of objects, differing in size and texture. The learning- and test situations are very complex, making the recognition accuracy comparable to real-world situations.

When considering only the best matches, as done in other research, the current results seem poor, only 49% of the objects can be recognised from different viewpoints. This is of course in a highly complex real-world environment. The results are close to improvement as can be seen from taking the top-2 or top-5 results, resulting in 69.4% and 82.7% respectively. This shows that the system described in this work can be applied in real-world situations and that with small modifications the recognition rates can be greatly improved.

Recognition rates of other systems concerned with learning and recognising objects done on robotic systems where only the first match was considered, vary: 63%-100% (Andreasson and Duckett, 2003), 90% (Kootstra et al., 2007), 92%-95% (Zickler and Veloso, 2006) and 66%-100% (Ekvall et al., 2006) in other research. However, each research uses a different set of objects and environments and each of these ran their experiments on different robot architectures, making comparison of these results hard, if not impossible. A recurring factor in the field of robotics is the use of a small dataset (2, 4, 7, and 9 objects for Andreasson and Duckett (2003), Ekvall et al. (2006), Kootstra et al. (2007), Zickler and Veloso (2006) respectively). The high recognition rates found in other literature may be attributed to that.

Improvements

Currently, objects are only recognised as present in the image, but it is also possible to detect the position and pose of the object. One method to estimate object position is that of Zickler and Veloso (2006). In this approach the orientation of each keypoint is stored during the learning stage, together with its relative position of the object centroid. During recognition each matching keypoint then votes for the location of the object centroid in that image. Keypoints are clustered based on these votes and in this way the object position is determined. The object pose can be determined by determining the best matching viewpoint from the original set of trainings viewpoints (Paletta and Pinz, 2000) or by applying a Hough transform to the matching keypoints (Lowe, 2004). The latter approach means that a greater accuracy can be achieved, as the matching is not limited to a certain viewpoint that was in the trainingsset, but it is accompanied with great computational costs (Lowe, 2004). A combination of these methods could thus be used, by improving the accuracy of the pose estimate given by a view recognition method using a Hough transform. Also, for each method the default settings were used, instead where noted. These default values are not optimal for all cases, meaning that a further increase in recognition accuracy can be expected by optimizing these values (Moreels et al., 2007).

As described in section 3.2.1, the position in 3D space relative to the robot's starting point is stored in the object model for each keypoint. This information is not used during recognition, as it was out of the scope of this research. Recognition accuracy can be expected to increase by using such information (Lowe, 2004, Murphy-Chutorian and Triesch, 2005). An example use is to estimate size of the object by using the relational distance between keypoints and discarding object hypotheses that are not consistent with the distances between keypoints. As the results of experiment 4.1 show (see section 5.4.1), the correct match is in many cases in the top 5 of matches. An interesting hypothesis is that the accuracy of the method can be increased when discarding incorrect matches by looking at the relative distance between individual keypoints.

Object recognition is currently not real-time, but several improvements are possible. The simplest of these is to increase the processing power of the robot hardware. Another possibility would be to use less keypoints in the object descriptions, by applying clustering to the keypoints (Kootstra et al., 2007, Murphy-Chutorian and Triesch, 2005, Zickler and Veloso, 2006). Currently only one type of object representation is used, a representation based on SURF. Another improvement that can be made on the current approach is to store multiple types of object representations. By using two methods an object hypothesis generated by one method can be confirmed or discarded by the other method. By using a fast method for creating object-hypothesis and confirming them with a slower method not the entire image needs to be searched by the slower method (Ekvall et al., 2006).

Currently active exploration is only done during object learning, while the use of exploration is also beneficial during the recognition stage (Kootstra et al., 2007, Paletta and Pinz, 2000, Schneider et al., 2005). To aid in active exploration it would be useful to be able to locate objects on a world map. By creating a map, the certainty of an object's position can be gradually increased. Besides this, it is useful in robot planning (Ekvall et al., 2006).

6.5 Conclusion

The proposed system is an improvement over several other systems currently used in robotics. It successfully combines human-robot interaction with active vision for object learning. By using filtering

techniques the object models can be decreased in size tremendously, while the recognition rate is not decreased. The resulting object models can be used in a recognition method to successfully detect objects in difficult scenes. Because the methods are tested against a more realistic database of objects than is normally the case in robotics, the results can be seen as more representative of real-world situations.

Bibliography

Andreasson, H. and Duckett, T. (2003). Object recognition by a mobile robot using omni-directional vision. In *Scandinavian Conference on Artificial Intelligence: SCAI'03*. IOS Press.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localisation and mapping (SLAM): Part II-State of the art. *Robotics and Automation Magazine*, 13(3):108–117.

Ballard, D. (1991). Animate vision. *Artificial Intelligence Journal*, 48:57–86.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Lecture Notes in Computer Science*, 3951:404.

Boykov, Y. and Jolly, M. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In *International Conference on Computer Vision*, volume 1, pages 105–112. Vancouver, BC, Canada.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc.

Breazeal, C., Brooks, A., Gray, J., Hoffman, G., Kidd, C., Lee, H., Lieberman, J., Lockerd, A., and Mulanda, D. (2004). Humanoid robots as cooperative partners for people. *Int. Journal of Humanoid Robots*, 1(2):1–34.

Brooks, A. G. and Breazeal, C. (2006). Working with robots and objects: revisiting deictic reference for achieving spatial common ground. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 297–304, New York, NY, USA. ACM.

Bühler, K. (1934). *Sprachtheorie*. G. Fischer.

Craig, J. (1989). *Introduction to robotics: mechanics and control*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Deinzer, F., Denzler, J., and Niemann, H. (2000). Classifier Independent Viewpoint Selection for 3-D Object Recognition. *Mustererkennung*, 22:237–244.

Dutta Roy, S., Chaudhury, S., and Banerjee, S. (2004). Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446.

Ekvall, S., Jensfelt, P., and Kragic, D. (2006). Integrating active mobile robot object recognition and slam in natural environments. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation*.

Feldman, J. (2003). What is a visual object? *Trends in Cognitive Sciences*, 7(6):252–256.

Fitzpatrick, P. (2003). *From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot*. PhD thesis, Massachusetts Institute of Technology.

Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.

Ghidary, S., Nakata, Y., Saito, H., Hattori, M., and Takamori, T. (2002). Multi-modal interaction of human and home robot in the context of room map generation. *Autonomous Robots*, 13(2):169–184.

Gopalakrishnan, A., Greene, S., and Sekmen, A. (2005). Vision-based mobile robot learning and navigation. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 48–53.

Gremban, K. and Ikeuchi, K. (1994). Planning multiple observations for object recognition. *International Journal of Computer Vision*, 12(2):137–172.

Gullberg, M. (1999). Gestures in spatial descriptions. *Working Papers 47*, pages 87–97.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371.

Helping Hands association (2009). Helping hands association. `www.monkeyhelpers.org`.

Horn, L. and Ward, G. (2004). *The handbook of pragmatics*. Blackwell Publishing.

Johnson, A. E. and Hebert, M. (1998). Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16(9-10):635–651.

Kass, M., Witkin, A., and D., T. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Kazi, Z., Beitler, M., Salganicoff, M., Chen, S., Chester, D., and Foulds, R. (1995). Multimodal user supervised interface and intelligent control (MUSIIC) for assistive robots. In *1995 IJCAI workshop on Developing AI Applications for the Disabled*, pages 47–58.

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2.

Kemp, C. C., Anderson, C. D., Nguyen, H., Trevor, A. J., and Xu, Z. (2008). A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 241–248, New York, NY, USA. ACM.

Kirstein, C. and Mueller, H. (1998). Interaction with a projection screen using a camera-tracked laser pointer. In *MMM '98: Proceedings of the 1998 Conference on MultiMedia Modeling*, Washington, DC, USA. IEEE Computer Society.

Kolmogorov, V. and Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. *Lecture Notes in Computer Science*, pages 82–96.

Kootstra, G., Ypma, J., and de Boer, B. (2007). Exploring objects for recognition in the real word. In *IEEE International Conference on Robotics and Biomimetics, 2007. ROBIO 2007*, pages 429–434.

LeCun, Y., Huang, F., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2.

Leroux, C., Laffont, I., Biard, N., Schmutz, S., Desert, J. F., Chalubert, G., and Measson, Y. (2007). Robot grasping of unknown objects, description and validation of the function with quadriplegic people. In *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, pages 35–42.

Lomker, F. and Sagerer, G. (2002). A multimodal system for object learning. *Lecture Notes in Computer Science*, pages 490–497.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60.

Lundberg, C., Barck-Holst, C., Folkeson, J., and Christensen, H. I. (2003). Pda interface for a field robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2882–2888.

Malvar, H., He, L., and Cutler, R. (2004). High-quality linear interpolation for demosaicing of bayer-patterned color images. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04)*, volume 3.

Marr, D. and Nishihara, H. (1978). Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 269–294.

McNeill, D. and Arnheim, R. (1996). *Hand and mind: What gestures reveal about thought*. University of Chicago Press.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

Moller, B., Posch, S., Haasch, A., Fritsch, J., and Sagerer, G. (2005). Interactive object learning for robot companions using mosaic images. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2650–2655.

Moreels, Pierre, Perona, and Pietro (2007). Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision*, 73(3):263–284.

Murphy-Chutorian, E. and Triesch, J. (2005). Shared features for scalable appearance-based object recognition. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 16–21.

Nicolescu, M. and Mataric, M. (2001). Experience-based representation construction: learning from human and robot teachers. In *In Proc., IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*.

Olsen Jr, D. and Nielsen, T. (2001). Laser pointer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22. ACM New York, NY, USA.

Olson, C. (2001). Object-based vision and attention in primates. *Current Opinion in Neurobiology*, 11(2):171–179.

Paletta, L. and Pinz, A. (2000). Active object recognition by view integration and reinforcement learning.

Patel, S. and Abowd, G. (2003). A 2-way laser-assisted selection scheme for handhelds in a physical environment. *Lecture notes in computer science*, pages 200–207.

Pfeifer, R. and Scheier, C. (1999). *Understanding Intelligence*. MIT Press.

Prakash, S., Abhilash, R., and Das, S. (2006). SnakeCut. *ELCVIA. Electronic letters on computer vision and image analysis*, 6(3):13.

Quine, W. (1961). *From a logical point of view*. Harvard University Press Cambridge, Mass.

Roth, P. M., Donoser, M., and Bischof, H. (2006). On-line learning of unknown hand held objects via tracking. In *Proceedings of the Second International Cognitive Vision Workshop*.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut : interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314.

Saxena, A., Chung, S., and Ng, A. (2006). Learning depth from single monocular images. *Advances in Neural Information Processing Systems*, 18:1161.

Schneider, G., Wersing, H., Sendhoff, B., and Korner, E. (2005). Evolutionary optimization of a hierarchical object recognition model. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3):426–437.

Shapiro, L. and Stockman, G. (2001). *Computer vision*. Prentice Hall Upper Saddle River, NJ.

Shi, J. and Tomasi, C. (1994). Good features to track. In *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94.*, pages 593–600.

Smith, R. and Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56.

Steil, J., Götting, M., Wersing, H., Körner, E., and Ritter, H. (2007). Adaptive scene dependent filters for segmentation and online learning of visual objects. *Neurocomputing*, 70(7-9):1235–1246.

Takizawa, M., Makihara, Y., Shimada, N., Miura, J., and Shirai, Y. (2003). A service robot with interactive vision-object recognition using dialog with user. In *Proc. of 1st Int. Workshop on Language Understanding and Agents for Real World Interaction*, pages 16–23.

Tomko, S. and Rosenfeld, R. (2004). Speech graffiti vs. natural language: Assessing the user experience. In *in Proceedings of HLT / NAACL 2004*, pages 73–76.

Toptsis, I., Li, S., Wrede, B., and Fink, G. A. (2004). A multi-modal dialog system for a mobile robot. In *Proc. Int. Conf. on Spoken Language Processing*, volume 1, pages 273–276.

Treisman, A. (1986). Properties, parts, and objects. *Handbook of perception and human performance*, 2:1–70.

Wachsmuth, S., Fink, G. A., Kummert, F., and Sagerer, G. (2000). Using speech in visual object recognition. In *Mustererkennung 2000, 22. DAGM-Symposium Kiel, Informatik Aktuell*, pages 428–435. Springer.

Wersing, H., Kirstein, S., Gotting, M., Brandl, H., Dunn, M., Mikhailova, I., Goerick, C., Steil, J., Ritter, H., and Korner, E. (2007). Online learning of objects in a biologically motivated visual architecture. *International Journal of Neural Systems*, 17(4):219–230.

Wilson, A. and Pham, H. (2003). Pointing in intelligent environments with the worldcursor. In *INTERACT International Conference on Human-Computer Interaction*.

Wilson, A., Shafer, S., and Shafer, S. (2003). XWand: UI for intelligent spaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 545–552. ACM New York, NY, USA.

Yoon, K. and Rybski, P. (2007). Teaching procedural flow through dialog and demonstration. In *Proceedings of the 2007 International Conference on Intelligent Robots and Systems*, pages 807–814.

Zickler, S. and Veloso, M. (2006). Detection and localization of multiple objects. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 20–25.

Ziegler, J., Nickel, K., and Stiefelhagen, R. (2006). Tracking of the articulated upper body on multi-view stereo image sequences. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:774–781.

Appendix A

Stereo Vision

A.1 Short description

Stereo vision is the description of an array of techniques to to get 3D data from two cameras. This is done by correlating the images from the two cameras. Because both cameras perceive a slightly different view of the world, distances can be calculated.

For this research the Triclops stereo library is used. The Triclops library is a proprietary software library from Point Grey Research which can be used to calculate depth information and to rectify a camera image. The calculations on the stereo vision are all done on grey-valued images, but the library can handle color images as well, for instance for the rectification feature. The camera images have to be supplied to the Triclops library in a certain format, which is documented in the SDK ("Triclops Software Development Kit, Version 3.1"). The library does not provide ways to connect to the camera itself, instead this is done through the DC1394 library on Linux.

The depth calculation method supplied by Point Grey Research is rather simple. It relies solely on matching individual pixels from the left camera image to the right image using a mask. More sophisticated techniques are for instance to use instance Graph Cuts energy minimization (Kolmogorov and Zabih, 2002). Saxena et al. (2006) combine disparity calculation with cues supplied by a learning algorithm (cues are for instance gradients and defocus) and use these to estimate the depth of patches in the image instead of on individual pixels. These more sophisticated algorithms provide better depth information (e.g. more accurate depth and less pixels that could not be matched), but the advantage of this simple approach is that it is fast. For instance, Kolmogorov and Zabih (2002) report depth calculation times of 369 to 702 seconds. Even with this simple approach, depth estimation is already one of the most processor intensive computations in the current set up. As the depth estimation needs to run on a mobile robot, there are severe computational limitations.

A.2 Depth from Stereo Vision

The algorithm used is proprietary, but is discussed in the SDK. Before stereo processing is done, a low pass filter is applied to the images, they are then rectified and edge detection is applied. After this the disparity image is calculated, filtered for errors and optionally subpixel interpolation is done.

A.2.1 Preprocessing

First a low pass filter is applied, to make sure that the rectification operation does not to show anti-aliasing effects. The rectification process eliminates the distortion that results from the lens shape. In unrectified images such distortion can be seen by looking at lines that should be straight but which appear bent (see figure A.1). The rectification in the Triclops library also aligns the rows of the camera images. For the disparity calculation to work properly, it is important the images map onto each other properly, otherwise the resulting depth map would be wrong. After this an edge detection algorithm is

Original image Rectified

Figure A.1: The result of the rectification process

applied to the image, which is used to optimize matching. If edge detection was used the stereo-matching algorithm will make use of changes in brightness instead of absolute values of the pixels in the images. This is useful, as the autogain feature of the Bumblebee2 camera can differ per camera, while changes in intensity would remain constant in such a case. The Bumblebee2 camera used in this research appears to only adjust the gain to the camera values of the reference image so it would appear that this problem is less severe, however, there might be a lag in updating the gain of the other camera.

A.2.2 Depth from Disparity calculation

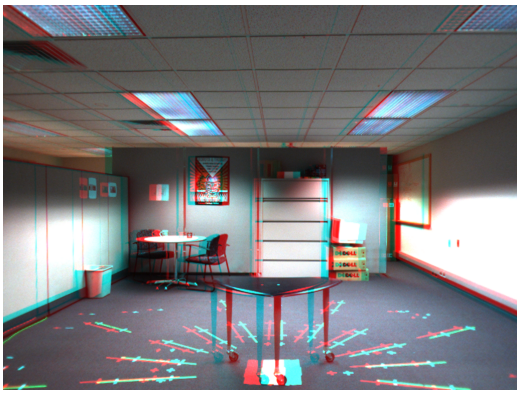
Stereo calculation is based on the difference of the images between the two cameras. Because each camera sees the world from a slightly different view, depth information can be calculated by comparing the two. Disparity calculation is done for each pixel in the reference image: around each pixel (x, y) in I_r a mask is taken (size $m \times m$). A mask of the same size is then put around a pixel in the other image on position $(x+d, y)$ in I_l. Because the camera images were aligned during rectification, the search for correct masks only has to be done over the horizontal plane. The disparity image I_d is then calculated as follows:

$$I_d(x,y) = \underset{d \in [d_{min}, \cdots, d_{max}]}{\operatorname{argmin}} \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} |I_r(x+i, y+j) - I_l(x+i+d, y+j)| \tag{A.1}$$

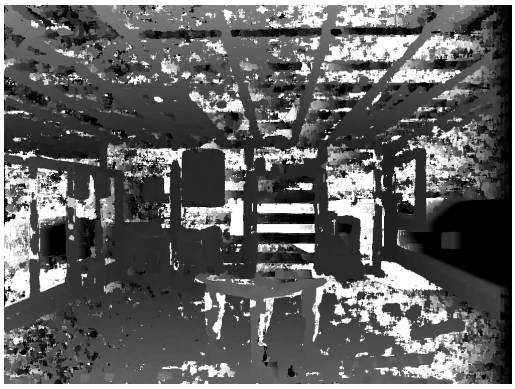
The minimum and maximum disparity (d_{min} and d_{max}) determine for which depth range values are found, a higher range means that more depth will be found but it will also increase computation time. The disparity image has the same size as the reference image and each position (x, y) in the disparity image I_d corresponds with position (x, y) in the reference image I_r. The disparity image can then be used to create a pointcloud, the distance in the real-world of each pixel can be calculated using the displacement d, the focal length of the cameras, the resolution of the CCD sensor and the displacement between the two cameras. The Triclops library does this automatically. Stereo matching might not always be possible, for instance in large areas without texture. In such a case a matching error occurs. The results of a stereo matching procedure can be seen in figure A.2, where matching errors are shown as white pixels.

Error correction

Especially in large errors without texture, the disparity information cannot be calculated. This leads to stereo errors in the pointcloud, pixels from which the position in 3D space is unknown. An additional averaging method was introduced, which can average surrounding depth information in the case of a matching error. The method takes the average position of all non-error positions in the pointcloud within a mask, whose size can be varied.



Merged left and right image



Resulting disparity image (mask $m = 15$)

Figure A.2: The result of stereo matching

A.3 Camera Settings

A.3.1 White balance and Gain

The white balancing and gain are done automatically in the Bumblebee2 camera, but both can be set manually through the DC1394-library. For the research described in this thesis the setting was left to automatic.

A.3.2 Demosaicing Algorithm

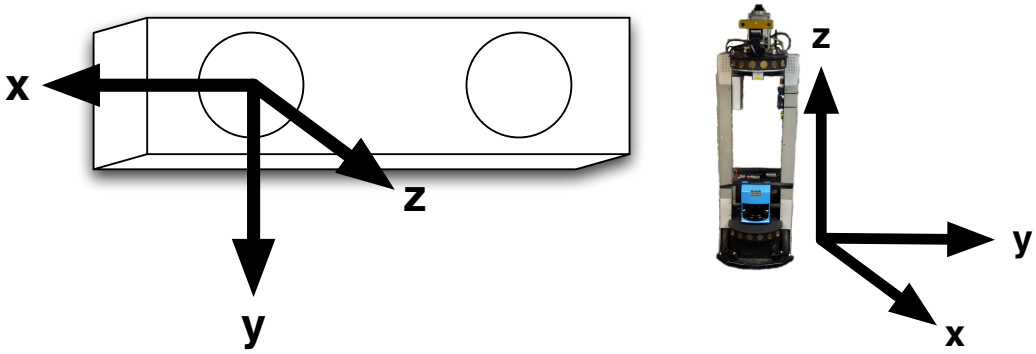
The Triclops library requires processed images, not the raw camera images. A digital camera is equipped with a CCD or CMOS sensor, both of which need to be equipped with a color filter array to record color images. This means that each pixel records the intensity of one of the primary colors (red, green or blue) instead of encoding a Red, Green, Blue (RGB) color value immediately. A common pattern used for the color filter is the Bayer Tile Pattern, which has twice as many green pixels as red or blue pixels. To get a complete color image, a demosaicing algorithm has to be used[1]. The Triclops library expects images to already be processed in this way, the DC1394-library provides several methods of doing this. The Bumblebee2 camera has Sony CCD chips which uses a Bayer Tile Pattern for its color filter. The difference between these algorithms can be seen easily in figure A.3.

A.4 Coordinate systems

The axis of the coordinate system of the Bumblebee 2 stereo camera and the Peoplebot differ, as can be seen in figure A.4(a) and (b). The camera uses a 'left-handed' coordinate scheme while the robot uses a right-handed one. Also, the camera returns its measurements relative to the right camera while all other sensors are configured towards a "robot origin" as defined by MobileRobots. The camera is mounted on a pan-tilt unit which is allowed to rotate over the z- and x-axis (with these axis defined in the robot axis system). To keep data from all sensors in the same coordinate system, the 3D positions that the stereo camera report have to be transformed. The pixels with their assigned 3D position are collectively called the pointcloud. This transformation is done with 3D affine transformations (Craig, 1989, Shapiro and Stockman, 2001). Such transformations are described in a matrix and can be applied to a vector describing a position by simply multiplication. Besides the x, y and z positions the position vector should also contain a scaling element. The transformation matrices contain a similar scaling factor in the fourth row.

[1] `http://en.wikipedia.org/wiki/Demosaicing`

Figure A.3: An advanced demosaicing method (HQLinear (Malvar et al., 2004)) and a quick, but less accurate demosaicing method (Nearest Neighbour)



(a) Axis system of the camera

(b) Axis system of the robot

Figure A.4: The two axis systems

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & d \\ 0 & 1 & 0 & d \\ 0 & 1 & 0 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Figure A.5: Example transformation, the point (x, y, z) is moved with distance d in all directions

The origin of the stereo camera is defined by Pt. Grey Research as the center of the "reference camera" on the front of the casing, which in this case is the right camera. The z-axis is defined as pointing outward from the camera center, the x-axis points to the right and the y-axis points downwards. The robot origin is defined by MobileRobots as being in the center of the wheel axis in the x,y plane. The original

coordinate system by MobileRobots did not provide a z-axis so it was defined as pointing upwards, having zero on ground level. The x-axis is defined as pointing outwards from the robot front and the y-axis is pointing to the left. A third origin was defined as an intermediary to ease transformations when the PTU (Pan Tilt Unit) was rotated. This origin is defined as the intersection of the pan- and tilt-axis in the PTU and has the robot axis system.

The first step in transforming points from the camera coordinate system to the robot system is redefining the axis of the point. This is simply done by rewriting the axis as in table A.1. After this the points are translated to the intermediary origin, a rotation transformation is done to account for rotation of the PTU and another translation is done to make the points relative to the robot origin.

Robot Axis	Camera Axis
X	Z
Y	-X
Z	-Y

Table A.1: Rewriting the axis system

The points are first translated to the intermediate origin, the translation matrix for this is A.6. From here the pointcloud is rotated with two rotational matrices, using the pan and tilt as reported by the PTU through the ARIA software architecture. It should be noted that these values are only valid if the PTU is not moving, this is due to a design choice in the ARIA software: the pan and tilt value are not polled directly from the PTU but instead those values are reported that were set last. The result of this is that even though the PTU is still moving it will report its final position. In this research this does not matter as the values are only read when the PTU is standing still. The PTU also introduces error as the actual pan and tilt positions are limited by the possible positions of the servo motors of the PTU, while the reported pan and tilt are not.

$$\begin{pmatrix} 1 & 0 & 0 & 0.0223 \\ 0 & 1 & 0 & -0.0600 \\ 0 & 0 & 1 & 0.0180 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure A.6: Translation matrix from the camera origin with respect to intermediary origin

The pointcloud is rotated around the pan axis first (using A.7) and over the tilt axis second (using A.8). This is because the pan-axis is defined as perpendicular to the tilt axis, which can be moved. The tilt axis is independent of the pan rotation. As the PTU can not roll, this axis is not present.

$$\begin{pmatrix} \cos\omega & -\sin\omega & 0 & 0 \\ \sin\omega & \cos\omega & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure A.7: Rotation matrix for pan (rotation around the z-axis)

$$\begin{pmatrix} \cos\tau & 0 & \sin\tau & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\tau & 0 & \cos\tau & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure A.8: Rotation matrix for tilt (rotation around the y-axis)

After the rotation matrices are applied, the points are still defined with respect to the intermediary origin. The final translation matrix needed to do translate the pointcloud to the robot origin can be found here A.9. The translation factors are all in meters and were determined with the help of the CAD drawings provided by the manufacturers and with careful measurements.

$$\begin{pmatrix} 1 & 0 & 0 & 0.0556 \\ 0 & 1 & 0 & -0.0112 \\ 0 & 0 & 1 & 1.212 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure A.9: Translation matrix from the intermediary origin with respect to robot origin

Appendix B

Measurements

B.1 Robot Template

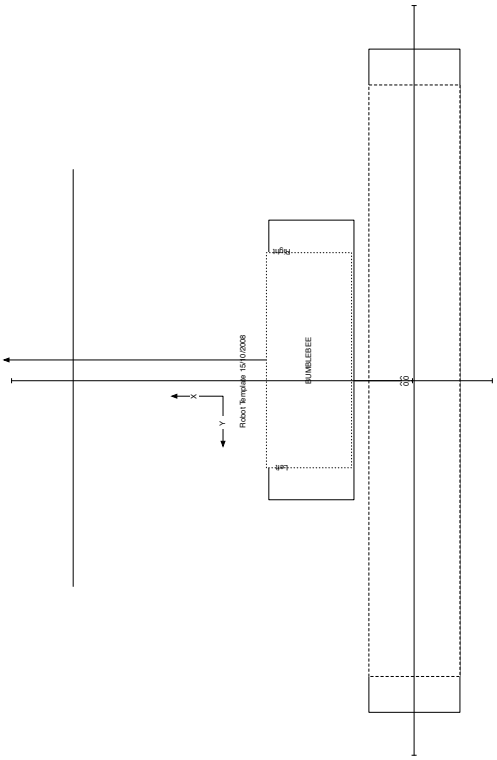
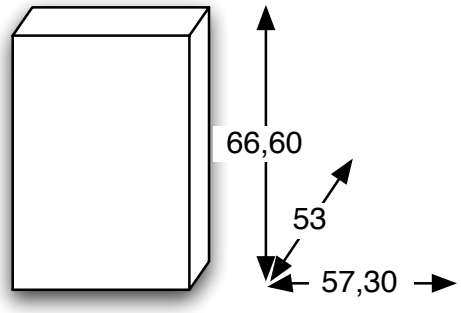
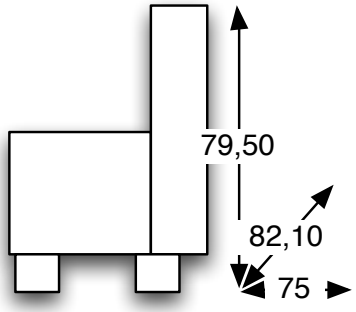


Figure B.1: By combining information from the robot blueprints (provided by MobileRobots), with careful measurements the transformation matrices described in section A.4 and this robot template were made. When enlarged to the correct format, this template can be used to position the robot precisely. Created together with Jaldert Rombouts

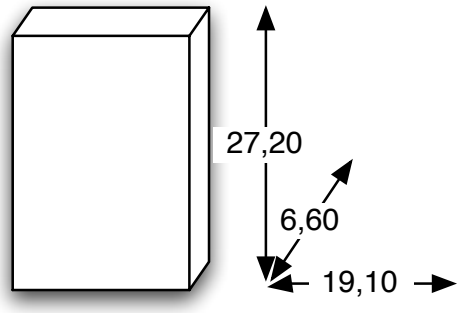
B.2 Object Learning and Validation Set



Big Box

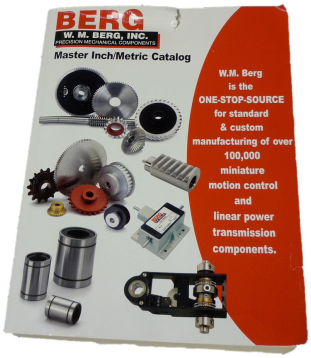


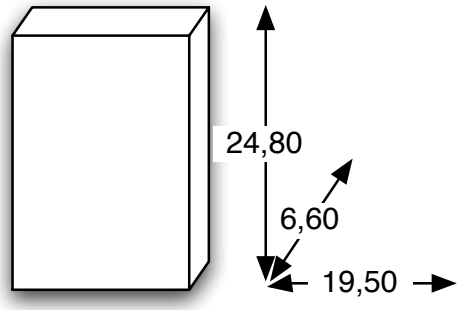
Big Chair



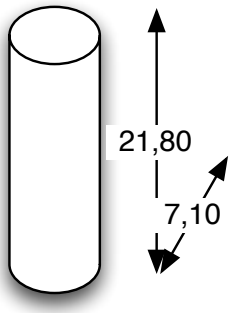
Book 1

Figure B.2: Objects in the learning and validation set.

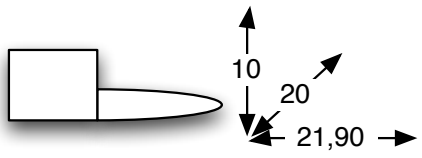




Book 2

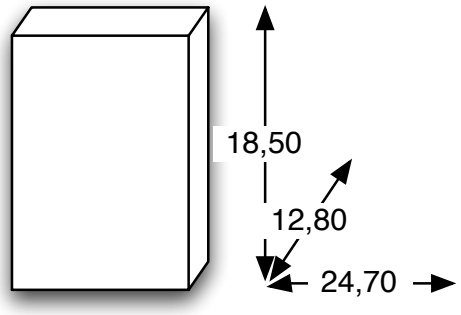


Bottle

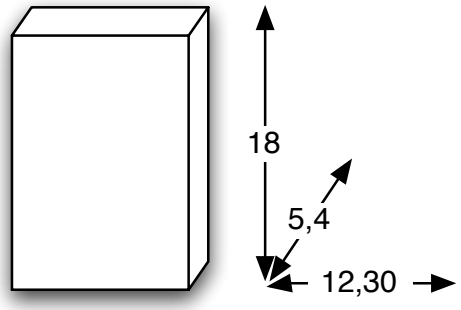


Cap

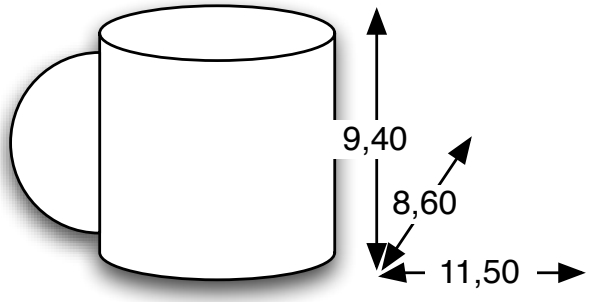
Figure B.3: Objects in the learning and validation set.



Choco Box 1

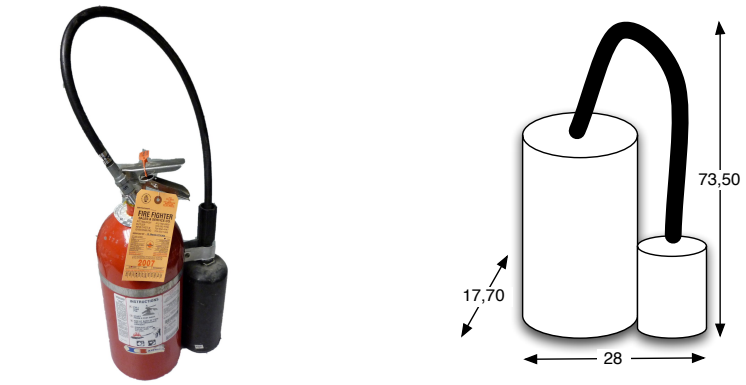


Choco Box 2

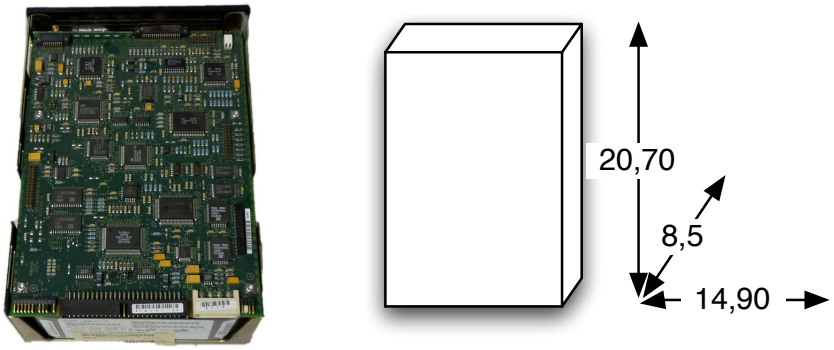


Cup

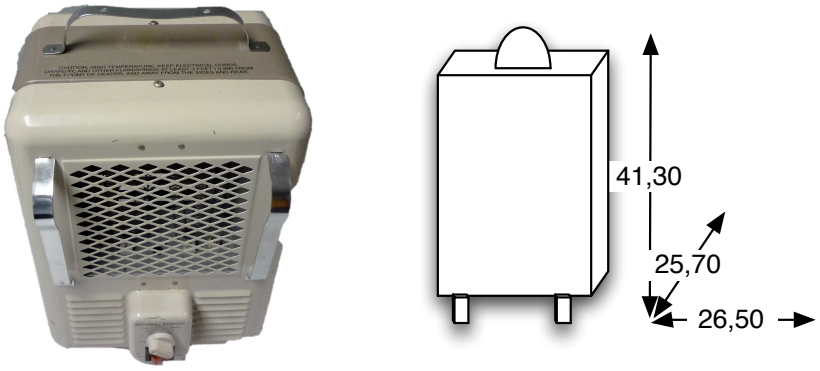
Figure B.4: Objects in the learning and validation set.



Fire Extinguisher

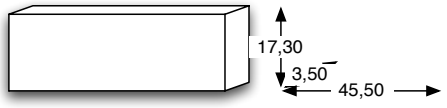


Hard Drive

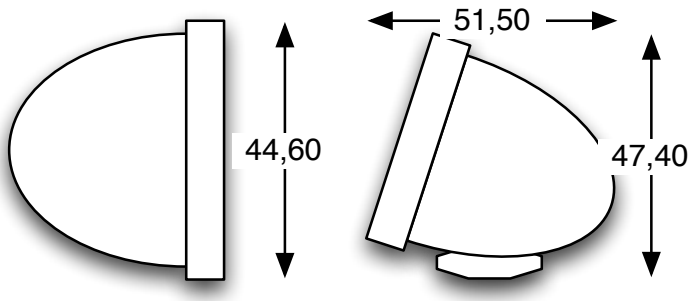


Heater

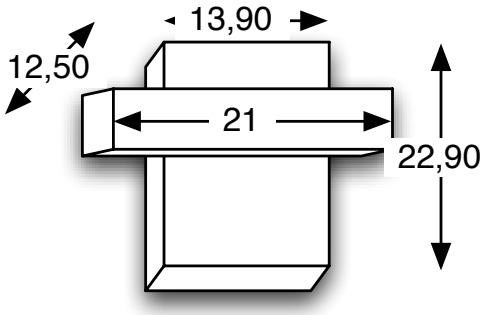
Figure B.5: Objects in the learning and validation set.



Keyboard

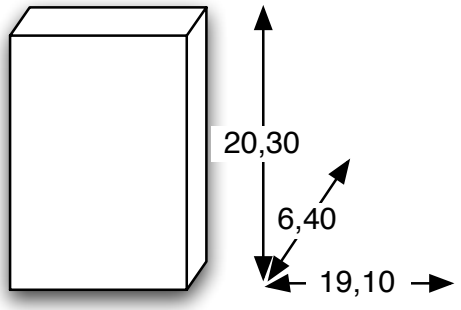


Monitor

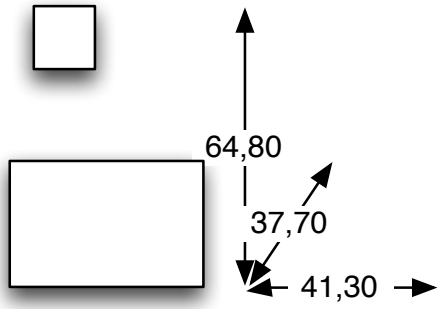


Phone

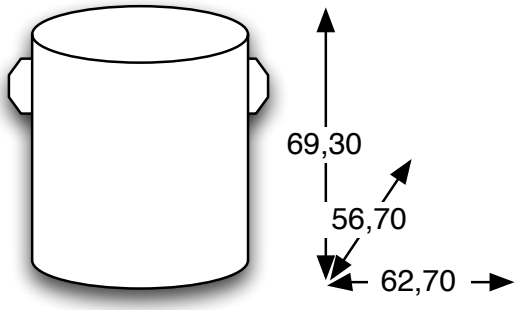
Figure B.6: Objects in the learning and validation set.



Pizza Box



Projector



Recycle Bin

Figure B.7: Objects in the learning and validation set.

Red Chair

Small Box

Table

Figure B.8: Objects in the learning and validation set.

Appendix C

Object Recognition Scenes

Figure C.1: *Scene 1 Trial 1*: Book 1, Pizza box, Big Chair, Chocolate box 2, Bottle

Figure C.2: *Scene 1 Trial 2*: Pizza box, Bottle, Book 1, Chocolate box 2

Figure C.3: *Scene 1 Trial 3*: Book 1, Chocolate box 2, Bottle, Pizza box

Figure C.4: *Scene 2 Trial 1*: Table, Book 2, Hard disk, Phone, Red chair, Big Chair, Cap

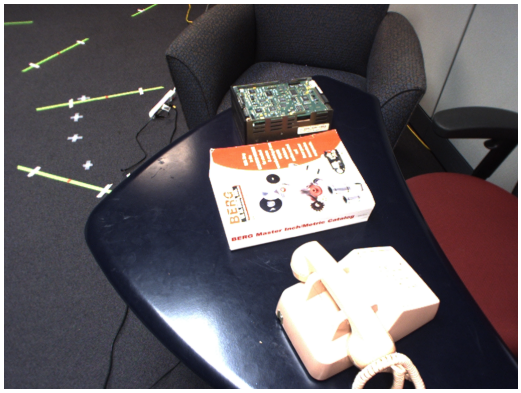


Figure C.5: *Scene 2 Trial 2*: Phone, Table, Book 2, Hard disk, Big Chair, Red Chair

Figure C.6: *Scene 3 Trial 1*: Small box, Big box, Fire extinguisher

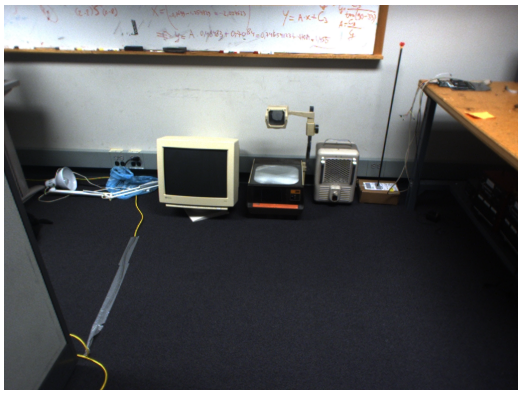


Figure C.7: *Scene 4 Trial 1*: Monitor, Heater, Projector

Figure C.8: *Scene 4 Trial 2*: Monitor, Heater, Projector

Figure C.9: *Scene 4 Trial 3*: Monitor, Heater, Projector

Figure C.10: *Scene 5 Trial 1*: Chocolate box 1, Chocolate box 2, Book 2, Hard disk, Cap, Bottle

Figure C.11: *Scene 6 Trial 1*: Monitor, Red Chair, Cap, Keyboard, Cup, Bottle, Small box, Phone, Book 1, Book 2

Figure C.12: *Scene 6 Trial 2*: Cap, Monitor, Keyboard, Cup, Bottle, Phone, Small box

Figure C.13: *Scene 6 Trial 3*: Monitor, Cap, Keyboard, Book 1, Book 2, Cup, Small box

Figure C.14: *Scene 6 Trial 4*: Red Chair, Heater, Monitor, Keyboard, Book 1, Book 2, Cup, Bottle Phone, Small box

Figure C.15: *Scene 7 Trial 1*: Big box, Big chair, Cup, Projector, Recycle bin, Fire extinguisher

Figure C.16: *Scene 7 Trial 2*: Cup, Big chair, Big box, Pizza box

Figure C.17: *Scene 7 Trial 3*: Projector, Big chair, Recycle bin, Fire extinguisher

Appendix D

List of acronyms

ALOI "Amsterdam Library of Image Objects" Geusebroek et al. (2005)

ARIA Advanced Robotics Interface for Applications

BIRON Bielefeld Robot Companion

CCD Charged-Coupled Device

EM Expectation Maximization

GLOH Gradient Localization and Orientation Histogram

GMM Gaussian Mixture Model

MUSIIC Multimodal User-Supervised Interface and Intelligent Control

PCA Principal Component Analysis

RGB Red, Green, Blue

RFCH Receptive Field Cooccurrence Histogram

SURF Speeded Up Robust Features

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization And Mapping

FN False Negative

FP False Positive

FPR False Positive Rate (also called recall)

TP True Positive

TPR True Positive Rate (also called precision)

TN True Negative